

**Q2 (a)** Define Information System? Who are the typical stakeholders in an information system? Discuss their roles?

**Answer:** An information system is an arrangement of people, data, processes, information presentation, and information technology that interacts to support and improve day-to-day operations in a business as well as support the problem solving and decision-making needs of management and users.

*The stakeholders for information systems can be broadly classified into six groups:*

1. *System owners* pay for the system to be built and maintained. They own the system, set priorities for the system, and determine policies for its use. In some cases, system owners may also be system users.
2. *System users* actually use the system to perform or support the work to be completed. System users define the business requirements and performance expectations for the system to be built.
3. *System designers* design the system to meet the user's requirements. In many cases, these specialists may also be system builders.
4. *System builders* construct, test, and deliver the system into operation.
5. *System analysts* facilitate the development of information system and computer applications by bridging the communications gap that exists between non technical system owners and users and technical system designers and builders.
6. *IT vendors and consultants* sell hardware, software, and services to businesses for incorporation into their information systems.

**(b)** Explain the different classes of Information System Applications?

**Answer:** There are five different classes of information system applications, which serve the needs of different users.

1. Transaction processing systems are information system applications that capture and process data about business transactions. It can either respond to business transactions such as orders, time cards or payments etc. or initiate transactions such as invoices, paychecks or receipts, or possibly both. Also, transaction processing can respond to both external events (such as processing orders from customers) and internal events (generating production orders).

One dimension of transaction processing system, data maintenance, provides for custodial updates to stored data. For example the system must provide for the ability to add and delete *customers* and *products*, as well as to change specific facts such as *customer address* etc. Business process redesign (BPR) is the study, analysis and redesign of fundamental business processes to reduce costs and / or improve value added to the business.

Examples of transaction processing system includes "Airline reservations", "hotel check-in / check-out", "Invoicing or billing", "Payroll" etc.

2. Management information system is an information system application that provides for management-oriented reporting. These reports are usually generated on a predetermined schedule and appear in a prearranged format. Management information systems can present detailed information, summary information, and exception information. Detailed information is used for operations management as well as regulatory requirements. Summary information consolidates raw data to quickly indicated trends and possible problems. Exception information filters data to report exceptions to some rule or criteria.  
Examples of management information system includes “Budget forecasting and analysis”, “Financial reporting”, “Inventory reporting”, “Sales forecasting” etc.
3. Decision support system is an information system application that provides its user with decision-oriented information whenever a decision-making situation arises. When applied to executive managers, these systems are sometimes called executive information system (EIS). A decision support system does not typically make decisions or solve problems. Decision support systems are concerned with providing useful information to support the decision process. In particular, decision supports systems are usually designed to support unstructured decisions, i.e. those decision-making situations that cannot be predicted. In general, a DSS provides one or more of the following types of support to the decision maker:
  - Identification of problems or decision-making opportunities.
  - Identification of possible solutions or decisions.
  - Access to information needed to solve a problem or make a decision.
  - Analysis of possible decisions or of variables that affect a decision. Sometimes this is called “what if” analysis.
  - Simulation of possible solutions and their likely results.
4. Expert systems are an extension of the decision support system. An expert system is a programmed decision-making information system that captures and reproduces the knowledge and expertise of an expert problem solver or decision maker and then simulates the thinking or actions of that expert. These expert systems often possess knowledge and expertise that cannot be easily be duplicated or replaced in all organizations. Expert systems are implemented with artificial intelligence technology that captures, stores, and provides access to the reasoning of the experts. Experts system requires data and information but is unique in their requirement of storing rules (called heuristics) that simulates the reasoning of the experts who use data and information.
5. Office automation system is more than word processing and spreadsheet applications. Office automation systems support the wide range of business office activities that provide for improved workflow and communications between workers, regardless of whether or not those workers are located in the same office. Office automation system is more concerned with getting all relevant

information to those who need it. Office automation functions include word processing, electronic messages, work group computing facsimile processing etc. Office automation systems are designed to support both individuals and work groups. Personal information systems are those systems that are designed to meet the needs of a single user. They are designed to boost an individual's productivity. Work group information systems are those designed to meet the needs of a work group. They are designed to boost the group's productivity.

**Q3 (a)** what are the basic ideas of RAD? Give some advantages and disadvantages of RAD approach?

**Answer:** Rapid application development (RAD) techniques emphasize extensive user involvement in the rapid and evolutionary construction of working prototypes of a system to accelerate the system development process. The basic ideas of RAD are as follows:

- To more actively involve system users in the analysis, design, and construction activities.
- To organize systems development into a series of focused, intense workshops jointly involving System owners, Users, Analysts, Designers, and Builders.
- To accelerate the requirements analysis and design phases through an iterative construction approach.
- To reduce the amount of time that passes before the users begin to see a working system.

Advantages of RAD approach:

- It is useful for projects in which the user requirements are uncertain or imprecise.
- It encourages active user and management participation. This increases end-user enthusiasm for the project.
- Projects have higher visibility and support because of the extensive user involvement throughout the process.
- Users and management see working, software-based solutions more rapidly than they do in model-driven development.
- Errors and omissions tend to be detected earlier in prototypes than in system models.
- Testing and training are natural by-products of the underlying prototyping approach.
- The iterative approach is a more natural process because change is an expected factor during development.

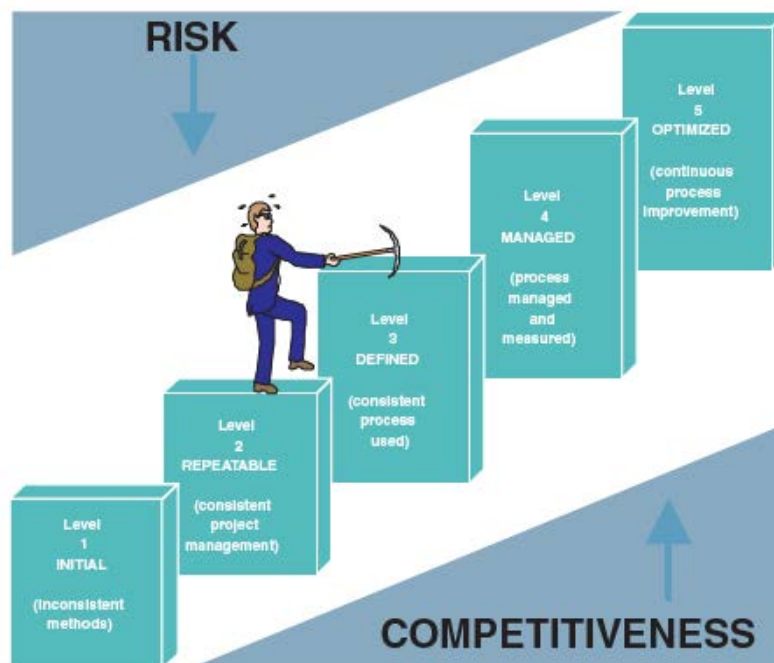
Disadvantages of RAD approach:

- RAD prototypes can easily solve the wrong problems since problem analysis is abbreviated or ignored.
- A RAD-based prototype may discourage analysts from considering other, more worthy technical alternatives.

- Sometimes it is best to throw a prototype away, but stakeholders are often reluctant to do so because they see this as a loss of time and effort in the current product.
- The emphasis on speed can adversely impact quality because of ill-advised shortcuts through the methodology.

(b) Describe Capability Maturity Model (CMM) for quality management?

**Answer:** Capability Maturity Model (CMM) a standardized framework for assessing the maturity level of an organization's information systems development and management processes and products. It consists of five levels of maturity.



The Capability Maturity Model (CMM)

- Level 1 – Initial: this is sometimes called anarchy or chaos. At this level, system development projects follow no consistent process. Each development team uses its own tools and methods. Success or failure is usually a function of the skill and experience of the team. The process is unpredictable and not repeatable. A project typically encounters many crises and is frequently over budget and behind schedule. Document is not consistent from one project to the next, thus creating problems for those who must maintain a system over its lifetime. Almost all organizations start at Level 1.
- Level 2 – Repeatable: At this level, project management processes and practices established to track project costs, schedules, and functionality. The focus is on project management. A system development process is always followed, but it

may vary from project to project. Success or failure is still a function of the skill and experience of the project team. Effective project management practices lay the foundation for standardized processes in the next level.

- Level 3 – Defined: In this level, a standard system development process, called as methodology, is purchased or developed. All projects use a tailored version of this process to develop and maintain information systems and software. As a result of using the standardized process for all projects, and each project results in consistent and high-quality documentation and deliverables. The process is stable, predictable, and repeatable.
- Level 4 – Managed: In this level, measurable goals for quality and productivity are established. Detailed measures of the standard system development process and product are routinely collected and stored in a database. There is an effort to improve individual project management based on this collected data. Thus, management seeks to become more proactive than reactive to system development problems such as, cost overruns, scope creep, schedule delays etc. Even when a project encountered unexpected problems or issues, the problem can be adjusted based on predictable and measurable impacts.
- Level 5 – Optimising: In this level, the standardized system development process is continuously monitored and improved based on measures and data analysis established in level 4. This can include changing the technology and best practices used to perform activities required in the standard system development process, as well as adjusting the process itself. Lessons learned are shared across the organization, with a special emphasis on eliminating inefficiencies in the system development process while sustaining quality.

**Q4 (a)** what is Rapid Architected Analysis? Describe the two different techniques for applying rapid architected analysis? Give examples for both techniques?

**Answer:** Rapid Architected Analysis is an accelerated analysis approach that also builds system models. Rapid architecture analysis is made possible by reverse-engineering technology that is included in many automated tools such as CASE and programming languages. Reverse engineering tools generate system models from existing software applications or system prototypes. The resulting system models can then be edited and improved by system analysts and users to provide a blueprint for a new and improved system. It should be apparent that rapid architected analysis is a blending of model-driven and accelerated analysis approaches.

There are two different techniques for applying rapid architected analysis:

- (i) Most systems have already been automated to some degree and exist as legacy information systems. Many CASE tools can read the underlying database



structures and / or application programs and reverse engineer them into various system models. These models serve as a point of departure for defining model-driven user requirements analysis.

- (ii) If prototypes have been built into tools like Microsoft Access or Visual Basic, those prototypes can sometimes be reverse engineered into their equivalent system models. The system models usually better lend themselves to analysing the user's requirements for consistency, completeness, stability, scalability, and flexibility to future change. Also, the system models can frequently be forward engineered by same CASE tools and application development environments (ADEs) into database and application templates or skeletons that will use more robust enterprise-level database and programming technology.

**(b)** What is the purpose of Scope Definition Phase? List the five tasks you do in the scope definition phase?

**Answer:** The scope definition phase is the first phase of the classic systems development process. This phase might be called the preliminary investigation phase, initial study phase, survey phase, or planning phase. The scope definition phase answers the question, "Is this project worth looking at?" To answer this question, we must define the scope of the project and the perceived problems, opportunities, and directives that triggered the project. The scope definition phase must also establish the project plan in terms of scale, development strategy, schedule, resource requirements, and budget. This phase is concerned primarily with the system owner's view of the existing system and the problems or opportunities that triggered the interest. System owners tend to be concerned with the big picture, not details and they determine whether resources can and will be committed to the project.

The final deliverable for the preliminary investigation phase is completion of a "project charter". A project charter defines the project scope, plan, methodology, standards, and so on. Completion of the project charter represents the first milestone in a project. The scope definition phase is intended to be quick. The entire phase should not exceed two or three days for most project.

The scope definition phase typically includes the following five tasks:

- (i) Identify baseline problems and opportunities.
- (ii) Negotiate baseline scope.
- (iii) Assess baseline project worthiness.
- (iv) Develop baseline schedule and budget.
- (v) Communicate the project plan.

(c) Write short note on BPR?

**Answer:** Business process redesign (BPR) the application of the systems analysis methods to the goal of dramatically changing and improving the fundamental business processes of an organization, independent of information system. The interest in BPR was driven by the discovery that most current information systems and applications have merely automated existing and inefficient business processes. BPR is one of the many types of projects triggered by the trends called as total quality management (TQM) and continuous process improvement (CPI).

Some BPR projects focus on all business processes, regardless of their automation. Each business process is thoroughly studied and analysed for bottlenecks, value returned, and opportunities for elimination or streamlining. Process models, such as data flow diagrams, help organizations visualize their processes. Once the business processes have been redesigned, most BPR projects conclude by examining how information technology might best be applied to the improved business processes. This may create new information system and application development projects to implement or support the new business processes.

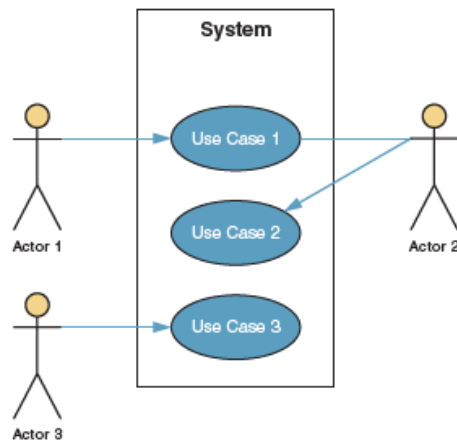
BPR is also applied within the context of information system development projects. It is not uncommon for information system projects to include a study of existing business to identify problems, and inefficiencies that can be addressed in requirements for new and improved information systems and computer applications. BPR has also become common in information system projects that will be based on the purchase and integration of the commercial off-the-shelf (COTS) software.

**Q5 (a)** Briefly describe the Use-case modeling techniques? List the benefits that are provided by use-case modeling?

**Answer:** Use-case modeling is an approach that facilitates usage-centered development. It has its roots in object-oriented modeling, but it has gained popularity in nonobject development environments. Use-case modeling was originally conceived by Dr. Ivar Jacobson in 1986 and gained popularity in 1992. Dr. Jacobson used use-case modeling as the framework for his objectory methodology, which he successfully used for developing object-oriented information systems. Use-case modeling has proved to be valuable aid in meeting the challenges of determining what a system is required to do from a user and stakeholder perspective. It is widely recognized as a best practice for defining, documenting, and understanding of an information system's functional requirements. Using use-case modeling facilitates and encourages user involvement, which is one of the primary critical success factors for ensuring project success.

There are two primary artifacts involved when performing use-case modeling. The first is the use-case diagram, which graphically depicts the system as a collection of use cases, actors (users), and their relationships. The second artifact, called the use-case narrative, fills in details of each business event and specifies how the users interact with the system during that event.

Use-case modeling identifies and describes the system functions by using a tool called **use cases**. Use cases describe the system functions from the perspective of the external users and in a manner and terminology they understand. Use cases are represented graphically by a horizontal ellipse with the name of the case appearing above, below, or inside the ellipse. A use case represents a single goal of the system and describes a sequence of activities and user interactions in trying to accomplish the goal. The creation of use cases has proved to be an excellent technique to better understand and document system requirements.



A sample Use Case Model diagram

#### Use-case modeling provides the following benefits:

- Provides a tool for capturing functional requirements.
- Assists in decomposing system scope into more manageable pieces.
- Provides a means of communicating with users and other stakeholders concerning system functionality. Use cases present a common language that is easily understood by various stakeholders.
- Provides a means of identifying, assigning, tracking, controlling, and managing system development activities, especially incremental and iterative development.
- Provides an aid in estimating project scope, effort, and schedule.
- Provides a baseline for testing in terms defining test plans and test cases.
- Provides a baseline for user help systems and manuals as well as system development documentation.
- Provides a tool for requirements traceability.
- Provides a starting point for the identification of data objects or entities.
- Provides functional specifications for designing user and system interfaces.
- Provides a means of defining database access requirements in terms of adds, changes, deletes, and reads.
- Provides a framework for driving the system development project.



(b) Describe some criteria that makes a good Data model?

**Answer:** Following are the criterion that makes a good data model:

- A good data model is simple: As a general rule, the data attributes that describe any given entity should describe only that entity.
- A good model is essentially non redundant: This means that each data attribute, other than foreign keys, describes at most one entity.
- A good data model should be flexible and adaptable to future needs: In the absence of this criterion, we would tend to design databases to fulfil only today's business requirements. Then, when a new requirement becomes known, we can't easily change the databases without rewriting many or all of the programs that used those databases.

**Q6 (a)** Today many analysts and designers prefer prototyping a modern engineering based approach to design. Describe some advantages and disadvantages of this approach.

**Answer:** Advantages of Prototyping approach are as follows:

- Prototyping encourages and requires active end-user participation. This increases end-user morale and support for the project. End user's morale is enhanced because the system appears real to them.
- Iteration and changes are a natural consequence of systems development, i.e. end users tend to change their minds. Prototyping better fits this natural situation because it assumes that a prototype evolves, through iteration, into the required system.
- It has often been said that end users don't fully know their requirements until they see them implemented. If so, prototyping endorses this philosophy.
- Prototypes are an active, not passive, model that end users can see, touch, feel and experience.
- An approved prototype is a working equivalent to a paper design specification, with one exception; errors can be detected much earlier.
- Prototyping can increase creativity because it allows for quicker user feedback, which can lead to better solutions.
- Prototyping accelerates several phases of the life cycle, possibly by passing the programmer. In fact, prototyping consolidates parts of phases that normally occur one after the other.

Disadvantages of Prototyping approach are as follows:

- Prototyping does not negate the need for the systems analysis phases. A prototype can solve the wrong problems and opportunities just as easily as a conventionally developed system can.

- Numerous design issues are not addressed by prototyping. These issues can be inadvertently forgotten if you are not careful.
- Prototyping often leads to premature commitment to a design.
- During prototyping, the scope and complexity of the system can quickly expand beyond original plans. This can easily get out of control.
- Prototyping can reduce creativity in designs. The very nature of any implementation, for instance, a prototype of a report can prevent analysts, designers and end users from looking for better solutions.
- Prototypes often suffer from slower performance than their third-generation language counterparts.

(b) What is an activity diagram? Give the guidelines for constructing activity diagrams?

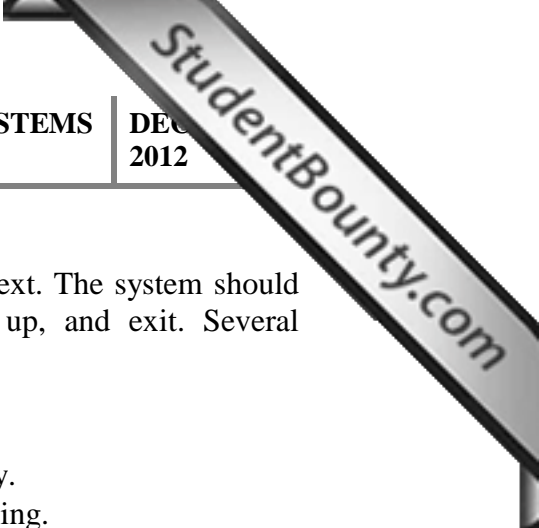
**Answer:** The activity diagram is used to model the process steps or activities of the system. They are similar to flowcharts in that they graphically depict the sequential flow of activities of either a business process or a use case. They are different from flowcharts in that they provide a mechanism to depict activities that occur in parallel. Because of this they are very useful to model actions that will be performed when an operation is executing as well as the results of those actions. Activity diagrams are flexible in that they can be used during both analysis and design. At least one activity diagram can be constructed for each use case. More than one can be constructed if the use case is long or contains complex logic. System analysts use activity diagrams to better understand the flow and sequencing of the use-case steps.

The following list presents the guidelines for constructing activity diagrams:

- Start with one initial node as a starting point.
- Add partitions if they are relevant to the analysis.
- Add an action for each major step of the use case or each major step an actor initiates.
- Add flows from each action to another action, a decision point, or an end point. For maximum precision of meaning, each action should have only one flow coming in and one flow going out, with all forks, joins, decisions, and merges shown explicitly.
- Add decisions where flows diverge with alternating routes. Be sure to bring them back together with a merge.
- Add forks and joins where activities are performed in parallel.
- End with a single notation for activity final.

**Q7 (a)** Based on the type of computer user, what are the important human engineering factors that can be incorporated into the system designs?

**Answer:** Given the type of user, a number of important human engineering factors should be incorporated into design:

- 
- i. The system user should always be aware of what to do next. The system should always provide instructions on how to proceed, back up, and exit. Several situations require some type of feedback:
    - Tell the user what the system expects right now.
    - Tell the user that data has been entered correctly.
    - Tell the user that data has not been entered correctly.
    - Explain to the user the reason for a delay in processing.
    - Tell the user that a task was completed or was not completed.
  - ii. The screen should be formatted so that the various types of information, instructions, and messages always appear in the same general display area. This way, the system user knows approximately where to look for specific information.
  - iii. Messages, instructions, or information should be displayed long enough to allow the system user to read them. Most experts recommend that important messages be displayed until the user acknowledges them.
  - iv. Use display attributes sparingly. Display attributes, such as blinking, highlighting, and reverse video can be distracting if overused.
  - v. Default values for fields and answers to be entered by the user should be specified. In windowing environments, valid values are frequently presented in a separate window or dialogue box as a scrollable region. The default value, if applicable, should usually be first and clearly highlighted.
  - vi. Anticipate the errors users might make. System users will make errors, even when given the most obvious instructions. If it is possible for the user to execute a dangerous action, let it be known.
  - vii. With respect to errors, a user should not be allowed to proceed without correcting an error. Instructions on how to correct the error should be displayed. The error can be highlighted with sound or color and then explained in a pop-up window or dialogue box.
  - viii. If the user does something that could be catastrophic, the keyboard should be locked to prevent any further input, and an instruction to call the analyst or technical support should be displayed.

**(b)** Distinguish between different types of computer users and design considerations for each?

**Answer:** System users can be broadly classified as either expert or novice and either nondiscretionary or discretionary.

An expert user is an experienced computer user who has spent considerable time using specific application programs. The use of a computer is usually considered nondiscretionary. In mainframe computing era, this was called a dedicated user. Expert users generally are comfortable with (but not necessarily experts in) the application's operating environment, such as Windows or Web browser. They have invested time in learning to use the computer. They will invest time in overcoming less-than-friendly user interfaces. In general, they have memorized routine operations to an extent that they

neither seek nor want excessive computer feedback and instructions. They want to be able to accomplish their task in as few actions and keystrokes as possible.

The **novice user**, also known as casual user, is less experienced computer user who will generally use a computer on a less frequent or even occasional basis. The use of a computer may be viewed as discretionary. The novice users need more help than expert users. Help takes many forms, including menus, dialogues, instructions, and help screens. Most managers despite their increasing computer literacy fall into novice category. They are paid to recognize and solve problems, exploit opportunities, and create plans and manage the vision, not to learn and use computers. Computers are considered tools by modern managers. When the need arises, they want to realize their benefit as quickly as possible and move on.

(c) What are the “commandments” offered by Galitz to solve the problems of user interface design?

**Answer:** Galitz offers the following overriding “commandments” of user interface design:

- Understand your users and tasks. This becomes increasingly difficult as we extend our information systems to implement business-to-consumer (B2C) and business-to-business (B2B) functionality using the Internet.
- Involve the user in interface design. Find out what the users like and dislike in their current applications. Involve them in screen design and dialogue from the beginning. This commandment is easily enabled with today’s PC database and rapid application development.
- Test the system on actual users. Observation and listening are the key skills here. After initial training, try to avoid excessive coaching and forcing users to learn the system. Instead, observe their actions and mistakes, and listen to their comments and questions to better understand their interaction with the user interface.
- Practice iterative design. The first user interface will probably be unsatisfactory. Expect any user interface design to go through multiple design iterations and testing.

**Q8 (a)** Define visibility and explain its three levels?

**Answer:**

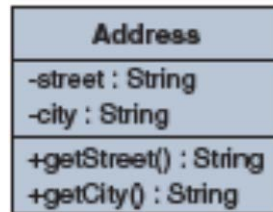
How attributes and methods are accessed by other classes is defined by visibility.

The UML provides three levels of visibility:

- Public – denoted by the symbol “+.”
- Protected – denoted by the symbol “#.”
- Private – denoted by the symbol “-.”

Public attributes can be accessed and public methods can be invoked by any other method in any other class. Protected attributes can be accessed and protected methods can be

invoked by any method in the class in which the attribute or method is defined or in subclasses of that class. Private attributes can be accessed and private methods can be invoked only by any method in the class in which the attribute or method is defined. If a method needs to be invoked in response to a message sent by another class, the method should be declared public. In most cases all attributes should be declared private to enforce encapsulation. The following figure depicts an example of denoting attribute and method visibility.



Visibility example

(b) In brief explain the following terms?

(i) Entity class

**Answer: Page No. 648, 656 of Textbook**

(ii) Strategy pattern class

**Answer: Page No. 669 of Textbook**

(iii) Deployment diagram class

**Answer: Page No. 693 of Textbook**

(c) What are the steps required to transform the class diagram prepared in OOA to a design class diagram?

**Answer:** The following steps are used to transform the class diagram prepared in OOA to a design class diagram:

1. Add design objects to diagram. The entity, interface, and control objects that were identified should be added to the diagram. Because of diagram space and readability considerations, only the major interface objects should be included.
2. Add attributes and attribute-type information to design objects. OO programming languages allow the common attribute types such as Integer, Date, Boolean, and String. OO languages also allow the definition complex attribute types such as Address, Social Security Number etc. this is a powerful feature for a developer.
3. Add attribute visibility. Attributes can be defined as public, protected, or private.
4. Add methods to design objects. Define methods to get and update the values of all the attributes of each object. These types of methods are commonly referred to as "setters" and "getters" methods. It is common to exclude these methods from the design class diagram in order to save space and make the diagram more readable, because they always exist by default. Also include methods to implement any



- previously identified responsibilities and behaviour, such as creating or deleting class instances or forming or breaking class associations.
5. Add method visibility. Methods can be defined as public, protected, or private.
  6. Add association navigability between classes. Add navigability arrows to unidirectional associations to indicate the direction messages are sent between source and target classes.
  7. Add dependency relationships. For any user interface class appearing on the diagram, draw a dependency line between it and the control object.

**Q9 (a)** Identify several system conversion strategies?

**Answer:**

The conversion plan may include one of the following commonly used installation strategies:

1. Abrupt cut-over – On a specific day, the old system is terminated and the new system is placed into operation. This is high risk approach because there may still be major problems that won't be uncovered until the system has been in operation for at least one business period. On the other hand, there are no transition costs. Abrupt cut-over may be necessary if, for instance, a government mandate or business policy become effective on a specific date and the system couldn't be implemented before that date.
2. Parallel conversion – Under this approach, both the old and new systems are operated for some time period. This ensures that all major problems in the new system have been solved before the old system is discarded. The final cut-over may be either abrupt or gradual, as portions of the new system are deemed adequate. This strategy minimizes the risk of major flaws in the new system causing irreparable harm to the business; however it also means the cost of running two systems over some period must be incurred.
3. Location conversion – When the same system will be used in numerous geographical locations, it is usually converted at one location first. As soon as that site has approved the system, it can be farmed to other sites. Other sites can be cut over abruptly because major errors have been fixed. Other sites benefit from the learning experiences of the first test site. The first production test site is often called a beta test site.
4. Staged conversion – Like location conversion, staged conversion is a variation on the abrupt and parallel conversions. A staged conversion is based on the version concept. Each successive version of the new system is converted as it is developed. Each version may be converted using the abrupt, parallel, or location strategy.

The conversion plan also typically includes a systems acceptance test plan. The system acceptance test is the final opportunity for end users, management, and

information systems operations management to accept or reject the system. A systems acceptance test is a final system test performed by the end users using real data over an extended time period. It is an extensive test that addresses three levels of acceptance testing – verification testing, validation testing, and audit testing.

**(b) What are the objectives of system maintenance?**

**Answer:** The fundamental objectives of system maintenance are:

1. To make predictable changes to existing programs to correct errors those were made during systems design or implementation.
2. To preserve those aspects of the programs that were correct and to avoid the possibility that “fixes” to programs cause other aspects of those programs to behave differently.
3. To avoid, as much as possible, degradation of system performance. Poor system maintenance can gradually erode system throughput and response time.
4. To complete the task as quickly as possible without sacrificing quality and reliability.

**(c) What are the different types of program restructuring?**

**Answer:** There are three distinct types of program restructuring:

1. Code reorganization restructures the modular organization and / or logic of the program. Logic may be restructured to eliminate control flows knots and reduce cycle complexity.
2. Code conversion translates the code from one language to another. Typically, this translation is from one language version to another. There is a debate on the usefulness of translators between different languages. If languages are sufficiently different, the translation may be very difficult.
3. Code slicing is the most intriguing program-reengineering option. Many programs contain components that could be factored out as subprograms. If factored out, they would be easier to maintain. More importantly, if factored out, they would be reusable. Code slicing cuts out a piece of a program to create a separate program or subprogram. This may sound easy, but it is not. Consider a COBOL program. The code you want to slice out may be located in many paragraphs and have dependent logic in many other paragraphs.

**TEXTBOOK: Systems Analysis and Design Methods, Jeffrey L Bentley, Seventh Edition, TMH, 2007**