Unit 8 CASE Tools for Systems Development

Lesson Structure

8.0 Objective

After going through this unit, you should be able to:

- understand the role of CASE tools and their use by the organizations;

- define various components of CASE tools;
- explain Visual CASE tools and know about some such commercial tools;
- understand the sophisticated CASE tools; and
- define Object Oriented CASE tools with their utility in development of software.

8.1 Introduction

A computer aided software engineering tool is also known as a CASE tool (Scotto). "The term CASE was originally coined by software company Nastec Corporation of Southfield Michigan in 1982 with their original integrated graphics and text editor GraphiText." (Arnold) Mark McMurttey et al describe it as "the automation of anything a human does to software."

Revolutionary changes are occurring in the traditional process of software system development. The normal SDLC process is often seen as inflexible and time consuming and expensive. Keeping in view of these limitations of SDLC process, the Computer Aided Software Engineering process has emerged to help organizations to develop systems and software. CASE involves using software packages called CASE tools to perform and automate many activities of system development life cycle. The CASE tools are helpful in business planning, project management, user interface design, database design and programming. Use of CASE tools makes computer-aided software development possible. In today's scenario where quick product delivery could be challenging task for the software vendor, CASE tools have come as helping hand to assist organization manage the software development process and automate certain processes of these activities. It is noteworthy that some capabilities of CASE are found in almost every modern software development tool. Some automatically design the user interface, few can auto generate codes, etc. It is difficult to imagine the life of a programmer without CASE tools.

CASE stands for **C**omputer **A**ided **S**oftware **E**ngineering CASE is the use of computer-based support in software engineering process. The support could be of any type like managerial, technical or administrative on any part of the software development process. All the software that helps in the process of software engineering can be termed as CASE tools.

8.2 Definition of CASE Tools

Computer-aided software engineering (CASE) is the scientific application of a set of tools and methods to a software system which is meant to result in high-quality, defect-free, and maintainable software products. It also refers to methods for the development of information systems together with automated tools that can be used in the software development process.

A CASE tool is a computer-based product aimed at supporting one or more software engineering activities within a software development process. Although, ideally a CASE tool should support all the activities of software engineering process starts from requirements analysis to designing, coding, testing, implementation and documentation, in reality, CASE tools often support one activity or at least a group of related activities. As software development activities become more complex and relatively unmanageable, there has been an awareness of the need for automated tools to help the software developer to accomplish this task. Initially, the focus was primarily on program support tools such as design of translators, compilers, assemblers, macro processors, and other tools. As computers became more powerful and the software that ran on them has grown larger and more complex, the support tools began to expand further. Some capabilities of CASE tools are also found in the common application development software. Large-scale use of computers has necessitated its maximum and efficient use and development of software for various activities of any organization. A software development effort can be viewed as a significant effort to design appropriate solutions, test and implement the solutions and finally documenting the solutions. In view of this, a wide range of support tools began to emerge to help the development team.

8.2.1 Use of CASE tools by organizations

There are number of CASE tools available to simplify various stages of Software Development Life Cycle such as Analysis tools, Design tools, Project management tools, Database Management tools, Documentation tools are to name a few.

Use of CASE tools accelerates the development of project to produce desired result and helps to uncover flaws before moving ahead with next stage in software development.

The following are some of the ways in which CASE tools are used by the Organization:

1. To facilitate single design methodology:

CASE tools help the organization to standardize the development process. It also facilitates coordinated development. Integration becomes easy as common methodology is adopted.

2. Rapid Application Development:

To improve the speed and quality of system development organizations use CASE tools.

3. Testing:

CASE tools help in improving the testing process through automated checking and simplified program maintenance.

4. Documentation:

In a traditional software development process, the quality of documentation at various stages depends on the individual. At various stages of SDLC CASE tools improve the quality and uniformity of documentation. It also ensures the completeness of the documentation.

5. Project Management:

It improves project management activity and to some extent automates various activities involved in project management.

6. Reduce the maintenance cost:

Use of CASE tools makes the software easy to maintain and hence reduce the maintenance costs.

7. Increase Productivity:

Automation of various activities of system development and management processes increases productivity of the development team.

8.2.2 Role of CASE Tools

CASE tools play a major role in the following activities:

- Project management
- Code generation
- Schema generation
- Reverse engineering
- Document generation
- OO analysis and design
- Data modeling
- Cost estimation
- Data dictionary
- User interface design
- Creation of meta-data for data warehouse
- Re-engineering
- Version control
- Software testing
- Project scheduling

CASE technology has resulted in significant improvements in quality and productivity. An ideal CASE tool should support all facets of system development like analysis, design, implementation, testing and maintenance. All aspects of software engineering process are not supported by today's CASE tool. Most of the CASE tools provide good support for data modeling, object oriented design and programming. Also, they moderately support testing and maintenance.

8.2.3 Advantages of CASE Tools

The following are some advantages of CASE tools:

- **Integrated development environment:**

Case tools produces system that more closely meet user needs and requirement.It provide unique user interface for the developer and analyst, automate time consuming and tedious activities like code generation.

l **Guidance in development:**

It provides common platform for all the developers and helps methodical system development.

l **Consistency between the model and documentation:**

Documentation is generated out of the model automatically leading to consistency between the model and documentation.

l **Longer Operational Life:**

CASE tools helps to develop the system having longer operational period.

l **Produces flexible System:**

The use of CASE tool result into the development of flexible System.

8.2.4 Disadvantages of CASE Tools

The following are some of the disadvantages of CASE tools:

- l Produces initial system that is more expensive to build and maintain.
- l Complex functionality.
- l Require more extensive and accurate definitions of user needs and requirements.
- l Many project management problems are not amenable to automation. Hence, CASE tools can't be used in such cases
- l Require training of maintenance staff.
- l May be difficult to use with the existing system.

8.3 Components of CASE

The activity that can be automated, whether partially or fully, depends on the CASE tools that are used. Most of the CASE tools generate a working model or prototype, which makes development process faster and easier.

8.3.1 Types of CASE Tools

The following are various types of CASE tools:

l **Planning and management tools:** These tools begin the development process with information planning and project management.

l **Analysis tools:** These tools ensure that business requirements are correctly captured during the analysis phase early in the development process. Analysis tools are used to check for incomplete, inconsistent or incorrect specifications.

l **Design toolset:** It provides detailed specification of the system.

l **Information integrator:** It integrates system specifications and checks them for consistency and completeness. It also records them in the CASE repository.

l **Code generator:** It automatically generates code specific to a language based on the system specification.

l **Database design toolset:** It suggests database design and generates system control information.

l **User interface generator:** It generates user interface based on system specification.

l **Report generator:** It generates reports based on specification.

All case tools are based on prototyping which is particularly useful when the user requirements are difficult to define. Large systems use traditional SDLC approach but part of the system can be prototyped. The prototype is then repeatedly refined till it becomes acceptable.

8.3.2 Classification of CASE Tools

CASE tools can be classified depending on the functionalities, these can be broadly classified into five generic categories:

l **Development tools:** These tools are interactive in nature. They are used for design support and code generation.

l **Front-end tools:** They support activities early in the life cycle of a software development process (planning, analysis and design). Examples are data flow diagram, data structure diagram, ER diagram, prototyping tools, etc.

l **Back-end tools:** They support activities later in the life cycle of a software development process (Implementation and maintenance). Examples are program flow chart, program editor, debugger, code generators etc.

l **Horizontal tools:** These tools are not specific to a particular life cycle step but are common across a number of life cycle steps e.g., Documentation tool.

l **Vertical tools:** These tools are specific to a life cycle.

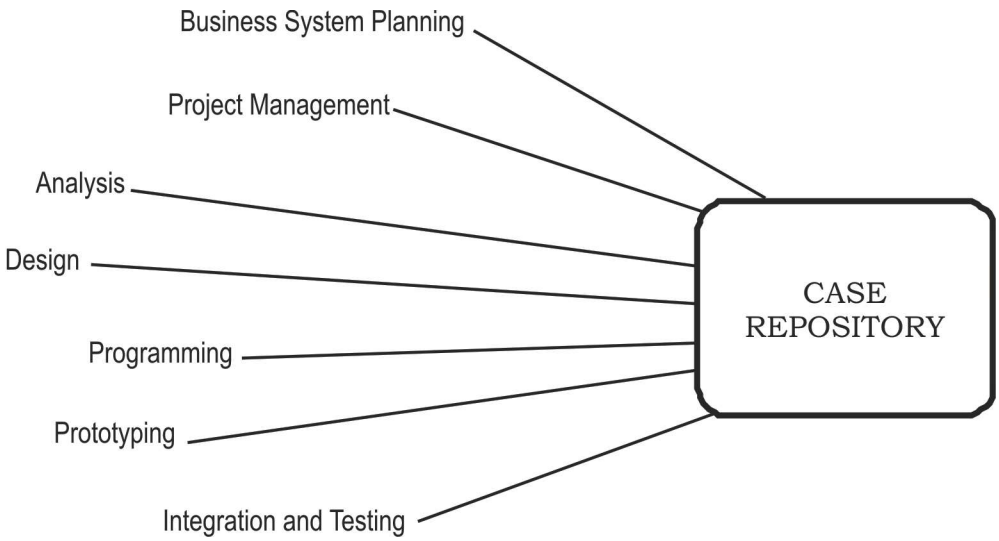


Figure 1 : Types of CASE tool

8.3.3 Reverse and Forward Engineering

We will discuss two important concept related to CASE tools namely:

(1) Reverse Engineering and

(2) Forward Engineering

Reverse engineering, in computer programming,is a technique used to analyze software in order to identify and understand the parts it is composed of. The usual reasons for reverse engineering a piece of software are to recreate the program, to build something similar to it, to exploit its weaknesses or strengthen its defenses.

Software companies with competing products reverse engineer their competitors' programs to find out where and how improvements can be made on their own products. Some companies use reverse engineering when they don't have similar products yet, to create products of their own.

Those who intend to build their own product based on an existing one often prefer reverse engineering over creating from scratch because once the parts and the dependencies are identified, the process of reconstructing tends to be much easier.

In the US, reverse engineering of software is protected by the fair use exception in copyright law.

Forward engineering is the process of building from a high-level model or concept to build in complexities and lower-level details. This type of engineering has different principles in various software and database processes.

Generally, forward engineering is important in IT because it represents the 'normal' development process. For example, building from a model into an implementation language. This will often result in loss of semantics, if models are more semantically detailed, or levels of abstraction.

Forward engineering is thus related to the term 'reverse engineering,' where there is an effort to build backward, from a coded set to a model, or to unravel the process of how something was put together.

It's crucial to note, though, that reverse engineering is also a term widely used in IT to describe attempts to take a software product or other technology apart and inspect how it works. In this type of contrast, forward engineering would be a logical 'forward-moving' design, where reverse engineering would be a form of creative deconstruction.

Some experts provide specific examples of forward engineering, including the use of abstract database models or templates into physical database tables. Other examples include a situation where developers or others make models or diagrams into concrete code classes, or specific code modules.

The figure given below explains forward and reverse engineering:

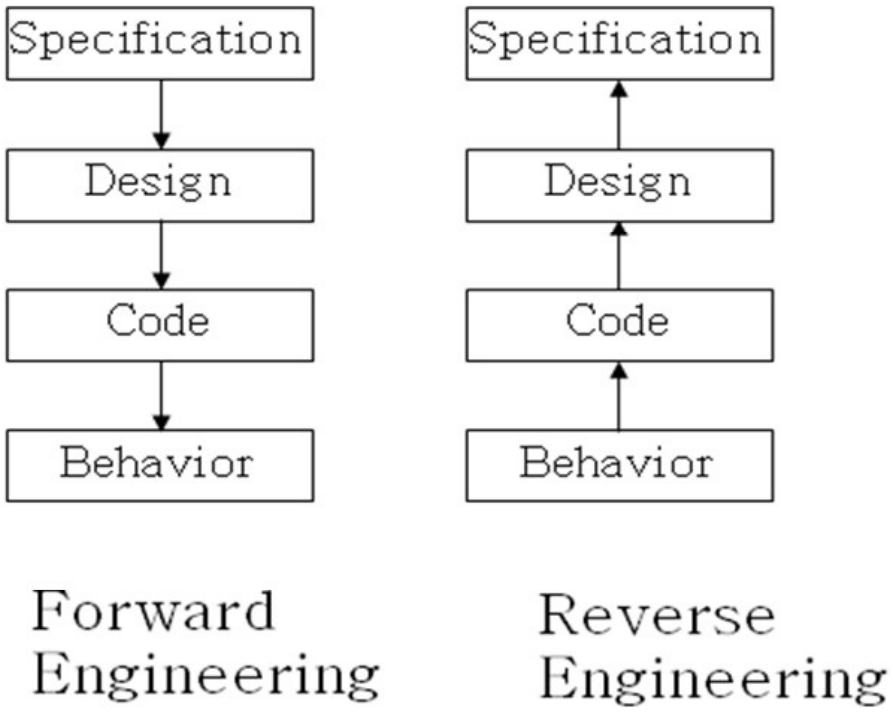


Figure 2

8.4 Visual and Emerging CASE Tools

Visual CASE tools is a tool that enable user to quickly create user interface and related skeleton code.

8.4.1 Traditional systems development and CASE based systems development :

The following are the features of Traditional Systems Development :

- Emphasis on coding and testing
- Paper-based specifications
- Manual coding of programs
- Manual documenting
- Intensive software testing
- Maintain code and documentation

The following are the feature of CASE-Based Systems Development :

- Emphasis on analysis and design
- Rapid interactive prototyping
- Automated code generation
- Automated documentation generation
- Automated design checking
- Maintain design specifications

8.4.2 CASE environment

Although individual CASE tools are useful, the true power of a tool set can be realized only when these set of tools are integrated into a common framework or environment. CASE tools are characterized by the stage or stages of software development life cycle on which they focus. Since different tools covering different stages share common information, it is required that they integrate through some central repository to have a consistent view of information associated with the software development artifacts. This central repository is usually a data dictionary containing the definition of all composite and elementary data items. Through the central repository all the CASE tools in a CASE environment share common information among themselves. Thus a CASE environment facilities the automation of the step-by-step methodologies for software development.

A typical CASE environment consists of a number of CASE tools and related components for supporting most or all phases of system development life cycle that operates on a common hardware and software platform.

CASE environment is not just a random amalgamation of CASE tools, it provides proper interaction between the CASE tools. One should concentrate less on which components should be chosen, and much more on how the selected components can be made to work together effectively.

The figure given below represents a typical CASE environment:

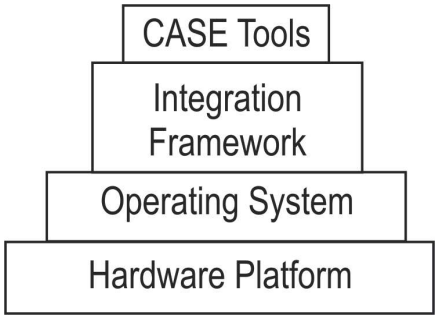


Figure 3

Examples of visual CASE Tools

- Oracle 2000 Designer
- Evergreen EasyCase
- Aonix: Software Through Pictures
- Popkin System Architect
- Cadre Teamwork
- ColdFusion
- Rational Rose
- Visual CASE
- Enterprise Architect.

Example of CASE tools

SELECT Enterprise : SELECT Enterprise is described as ' a dedicated modeling toolset for building scaleable client/server applications' . it provides integrated business process modeling, use case and object modeling based upon the UML development environment.

ORACLE Designer 2000 : ORACLE designer is one of the products offered by the ORACLE corporation. It is described as 'a model-based development toolset utilising a multi-user, repository-based environment' . The definitions of all model elements (Eg; entities, table definitions) are centrally stored in a repository, making them available to all members of the development team when and where they require them. Designer 2000 supports a number of complementary analysis techniques : process models, entity relationship diagrams, data flow diagrams etc.

Rational Rose : Rational Rose is a CASE toolset designed to support the UML development environment. The main benefits of rational Rose are described as multi-language support (C++, JAVA, VB, ADA), integrates with other industry standard environments and has a platform-independent development environment.

UML modeling: The Unified Modeling Language, or UML is mostly a graphical modeling language that is used to express designs. It is a standardized language in which artifacts and components of a software system can be specified. It is important to understand that UML simply describes a notation and not a process. It does not put forth a single method or process of design, but rather is a standardized tool that can be used in a design process.

The following are some of the tasks that can be performed by using CASE tools:

- UML modeling
- Code generation/construction for Visual C++, Visual Basic, C++, Ada, JAVA
- Database design
- Fully executable codes for C, C++,etc. across platforms
- Component testing.

Another CASE tool is Enterprise Architect (Parx System). This is an object oriented CASE tool for the entire software development life cycle. Its features are:

- Business process modeling
- Forward and reverse engineering
- Automation interface
- Support for C++, Java, VB, VB.Net, Delphi
- Project estimation tool

- Testing tool
- User interface design
- Requirement gathering
- Components model
- Deployment model

8.4.3 Emerging CASE Tools

Initially CASE tools were not very sophisticated in terms of the process they support. Now, integrated CASE tools have emerged to support the entire system engineering and software development process.

Integrated CASE (I-CASE)

It offers automated systems development environment that provides numerous tools to create diagrams, forms and reports.

- All tools share one common user interface.
- User has the feeling of working on one tool.
- Provide analysis, reporting and code generation facilities.
- Seamlessly shares and integrate data across and between tools.
- Repository is a central place to store information that is to be shared between various tools.

8.4.4 Creating documentation and reports CASE tools

A system requirement describes a condition or capability which a system must conform to, either directly from the user need or derived from or stated in a contract, standard, specification, or other formally imposed document

The object oriented CASE tools assist in documenting as well as in object oriented analysis and design. Object oriented CASE tools have capability to import graphics from other tools.

In the initial stage the system requirement is given shape in sketches with pencil and a paper. Then, it is used by the drawing tools and text editor available to create system diagram, class diagram and other diagrams. Document processor does it all to create a model document from these models.

System

Requirements

Rough sketch with pencil and paper

Figure 4 **: Process to create a model document**

8.4.5 Object Oriented Diagram Using CASE Tools

Object oriented CASE tools are similar to other CASE tools. These differ from others only in terms of their capability to create class diagrams and text specifications for reports. Object oriented CASE tools support an O-O methodology such as Rumbaugh's Object Management Technique (OMT).

The Object-Orientated Approach has the following feature

- The foundation of all development work is the object
- No new system models introduced at different stages
- Early models developed and refined through the development process
- An iterative design process

Examples of object oriented CASE tools

A large number of object oriented CASE tools are available in the market. These include:

Paradigm Plus from Protosoft,

Rational Rose from Rational and

WithClass from MicroGold Software.

The Object Oriented System consist of Object and Class. The Class is an object factory.The Object is defined by its class.All objects of the class have the same structure.

The class diagram is core to object-oriented design. It describes the types of objects in the system and the static relationships between them. The core element of the class diagram is the class. In an object-oriented system, classes are used to represent entities within the system. Entities are often related to real world objects.

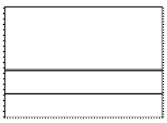
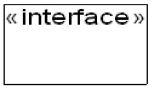
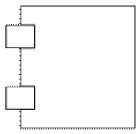
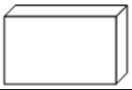
The Unified Modelling Language is :

- A notation or language for development.
- Not a development method.
- Set of diagrammatic techniques.
- Industry standard for modelling Object Oriented systems.
- UML Creators – Ivar Jacobson, Grady Booch, James Rumbaugh

The table given below discuss the principal UML Model

Model	View of the system
Use case	How the system interacts with its users.
Class	The data elements in the system and the relationships between them.
Interaction (sequence and collaboration)	How a use case affects all the objects that are involved in it.
State	How the different objects of a single class behave through all the use cases in which the class in involved.
Activity	The sequence of activities that make up a process.
Component	The different components of the system and the dependencies between them.
Deployment	The software and hardware elements of the system and the physical relationships between them.

The STRUCTURAL MODELING which is the core elements of UML is

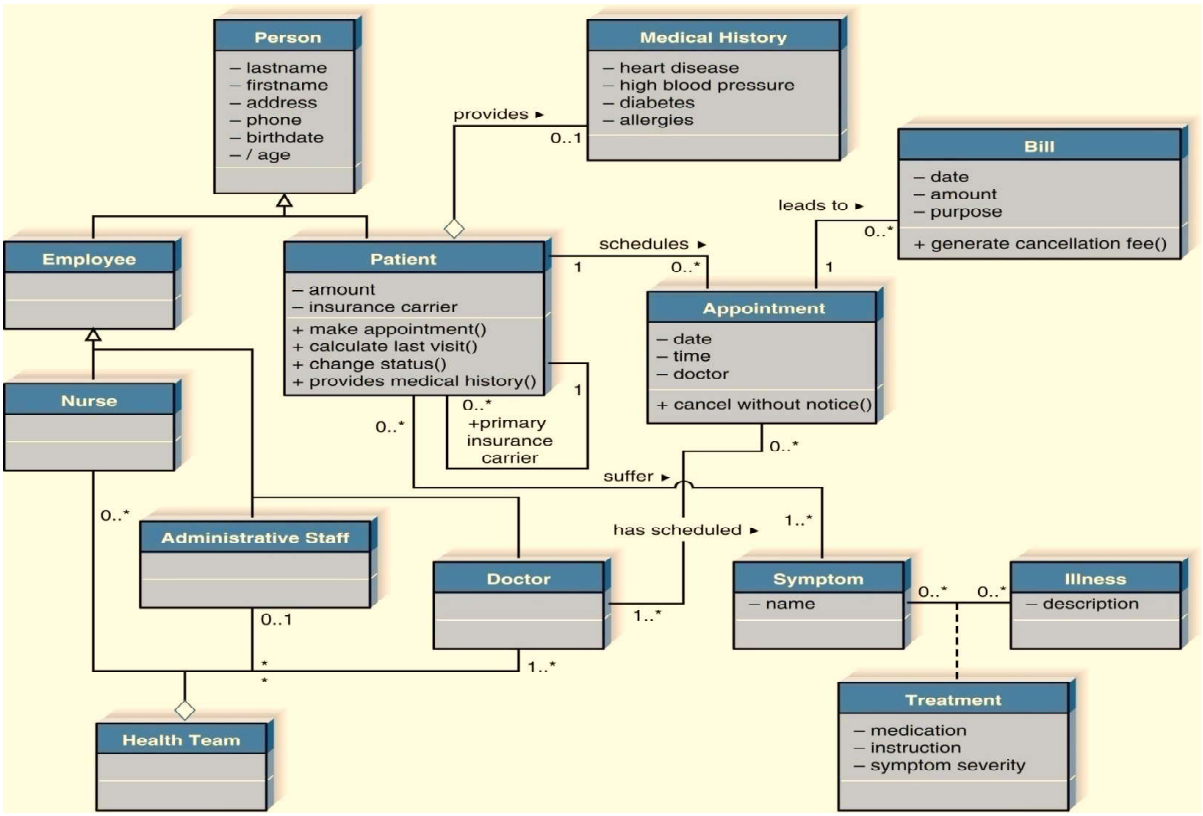
Construct	Description	Syntax
class	a description of a set of objects that share the same attributes, operations, methods, relationships and semantics.	
interface	a named set of operations that characterize the behavior of an element.	
Component	a modular, replaceable and significant part of a system that packages implementation and exposes a set of interfaces.	
node	a run-time physical object that represents a computational resource.	

Construct	Description	Syntax
association	a relationship between two or more classifiers that involves connections among their instances.	
aggregation	A special form of association that specifies a whole-part relationship between the aggregate (whole) and the component part.	
generalization	a taxonomic relationship between a more general and a more specific element.	
dependency	a relationship between two modeling elements, in which a change to one modeling element (the independent element) will affect the other modeling element (the dependent element).	

CLASS DIAGRAM

A class diagram is a static model that shows the classes and relationships among classes that remain constant in the system over time.

The diagram given below shows the class diagram for Hospital Management System



8.5 Summary

In this unit we came to know about CASE tool which is a Computer-Aided Software Engineering (CASE) tools.These are automated software packages that help to automate activities in the SDLC.CASE tools aim to enforce an engineering-type approach to the development of software systems.CASE tools range from simple diagramming tools to very sophisticated programs to document and automate most of the stages in the SDLC.These are used since the early 1990s.

A CASE tool provides iterative and interactive tools for various activities like user interface design and code generation. Modern CASE tools are aimed at supporting the entire activity of system development. Hence, CASE tools, known as I-CASE (Integrated CASE) has evolved. CASE tools are classified as front-end tools or back-end tools depending on which part of SDLC process they automate. Front-end tools automate initial part of SDLC process whereas back-end tools automate process that come later in SDLC life cycle.In this unit we have also discussed about Object Oriented Diagram Using CASE tool.

8.6 Questions for Exercise

1. What do you understand by the term CASE tool?
2. What are the primary objectives of a CASE tool?
3. What do you understand by the term CASE environment?
4. Differentiate between the characteristics of a CASE environment and a programming environment.
5. What are the main advantages of using CASE tools?
6. What do you mean by software reverse engineering and forward engineering?
7. Write short notes on:

   (i) Object Oriented diagram
   (ii) UML model
8. Define class and object.Draw Class diagram for Library Management System.

8.7 Suggested Reading

- Bennett, S., McRobb, S. and Farmer, R. Object-Oriented Systems Analysis and Design Using UML, 2nd Ed, London: McGraw-Hill, 2002.
- Joey George, J. Hoffer and Joseph Valacich; Modern Systems Analysis and Design,Third Edition, 2001, Pearson Education.

- Fowler, M. UML Distilled: a brief guide to the standard object modeling language, 2nd Ed, Reading Massachusetts: Addison-Wesley, 2000.
- James A. O'Brien; Introduction to Information Systems, An End user/Enterprise Perspective; Mc Graw Hill Edition; 1995

Reference Links

- http://www.ibm.com/developerworks
- http://linuxgazette.net
- http://www.techopedia.com

●●●