UNIT 10 CASE TOOLS FOR SYSTEMS DEVELOPMENT

Structure **Page Nos.**

10.0 INTRODUCTION

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.

10.1 OBJECTIVES

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

- know the role of CASE tools and their use by the organizations;
- know various components of CASE tools;
- explain Visual CASE tools and know about some such commercial tools;
- describe the sophisticated CASE tools; and
- describe Object Oriented CASE tools with their utility in development of software.

10.2 USE OF CASE TOOLS BY ORGANIZATIONS

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.

10.2.1 Definition of CASE Tools

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.

10.2.2 Use of CASE tools by organizations

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

- **To facilitate single design methodology:** CASE tools help organization to standardize the development process. It also facilitates coordinated development. Integration becomes easy as common methodology is adopted.
- **Rapid Application Development:** Organizations use CASE tools to improve the speed and quality of system development.
- **Testing:** CASE tools help to ease and improve testing process through automated checking and simplify program maintenance.
- **Documentation:** In traditional software development process, the quality of documentation at various stages depend on the individual. CASE tools improve the quality and uniformity of documentation at various stages of SDLC. It also ensures the completeness of the documentation.
- **Project Management:** It improves project management activity and to some extent automates various activities involved in project management.
- **Productivity and reduction of cost:** Use of CASE tools makes the software easy to maintain and hence reduce the maintenance costs. Automation of various activities of system development and management processes increases productivity of the development team.

10.2.3 Role of CASE Tools

CASE tools play a major role in the following activities:

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

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.

10.2.4 Advantages of CASE Tools

The following are some advantages of CASE tools:

- **Integrated development environment:** CASE tools provide unique user interface for the developer and analyst, automate time consuming and tedious activities like code generation.
- **Guidance in development:** It provides common platform for all the developers and helps methodical system development.
- **Consistency between the model and documentation:** Documentation is generated out of the model automatically leading to consistency between the model and documentation.

10.2.5 Disadvantages of CASE Tools

The following are some of the disadvantages of CASE tools:

- Complex functionality
- Many project management problems are not amenable to automation. Hence, CASE tools can't be used in such cases.

Check Your Progress 1

1. Systems analysts use automated software tools called _______ to develop information systems.
2. Organizations use CASE tools to increase the _______ and ________ of Systems development.
3. If a particular software development activity is not amenable to ________, then CASE tools cannot be used.

10.3 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.

10.3.1 Types of CASE Tools

The following are various types of CASE tools:

- **Planning and management tools:** Begin the development process with information planning and project management.
- **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.
- **Design toolset:** It provides detailed specification of the system.
- **Information integrator:** It integrates system specifications and checks them for consistency and completeness. It also records them in the CASE repository.
- **Code generator:** It automatically generates code specific to a language based on the system specification.
- **Database design toolset:** It suggests database design and generates system control information.
- **User interface generator:** It generates user interface based on system specification.
- **Report generator:** It generates reports based on specification

Figure 10.1 depicts various components of a typical CASE tool.

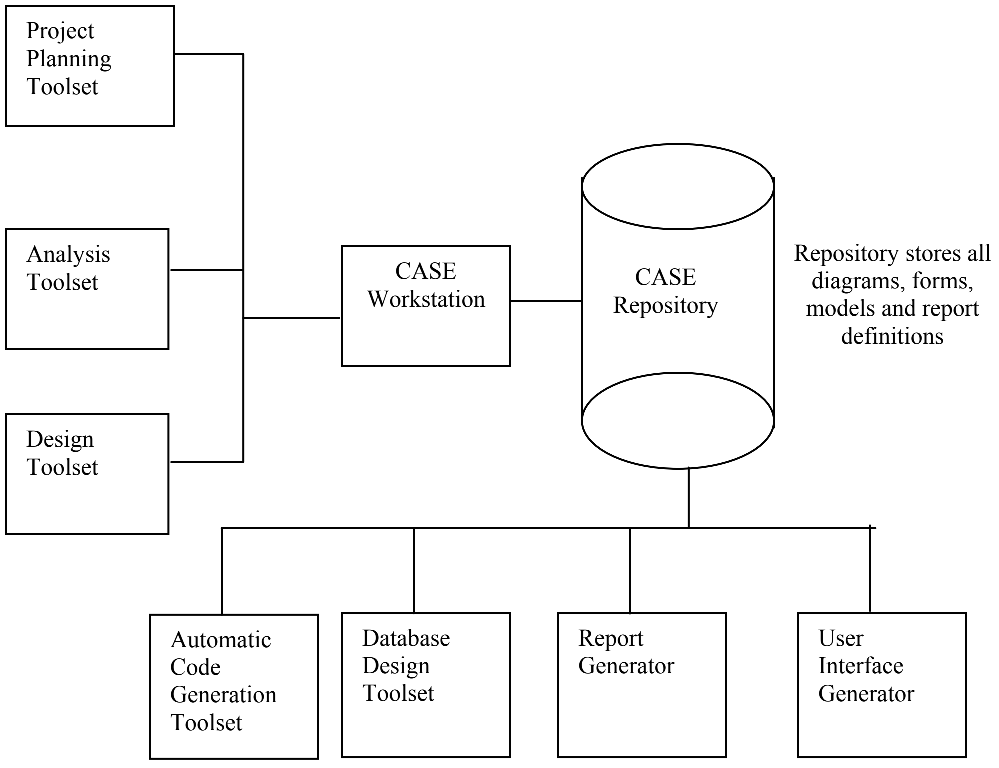

Figure 10.1: Components of a typical CASE tool

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.

10.3.2 Classification of CASE Tools

Figure 10.2 depicts various types of CASE tools.

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

- **Development tools:** These tools are interactive in nature. They are used for design support and code generation.
- **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.
- **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.
- **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.
- **Vertical tools:** These tools are specific to a life cycle.

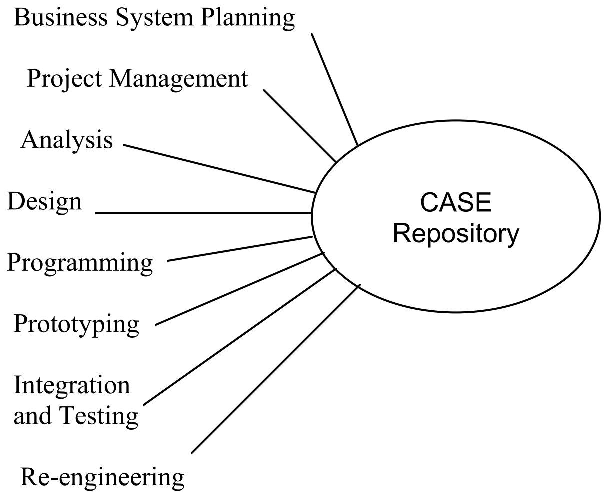

Figure 10.2: Types of CASE Tools

10.3.3 Reverse and Forward Engineering

We will discuss two important concept related to CASE tools namely, Forward Engineering and Reverse Engineering. Figure 10.3 depicts both Forward and Reverse Engineering.
Reverse engineering is the process of recreation of model based on existing code. First, the existing code is scanned to generate the model. Then the model can be fine tuned in accordance with requirements. Reverse engineering allows developers to create model for old systems, which were never modeled. It analyses existing software with purpose of understanding its design and specification. Reverse engineering tools read program source code and create graphical and textual representation of design.

Forward engineering is the process of generation of skeleton code out of the models. First step is to create the model for a system, then generate the relevant code for the model and then allow modification of this code in tune with the requirements.
Re-engineering means "restructuring and rewriting the legacy system or part of it without changing its original functionality". Re-engineering efforts make the software up-to-date to current technology and hence easy to maintain. The new system becomes restructured and re-documented. Re-engineering tools read program source code and interactively change and existing system to improve quality performance or maintainability.

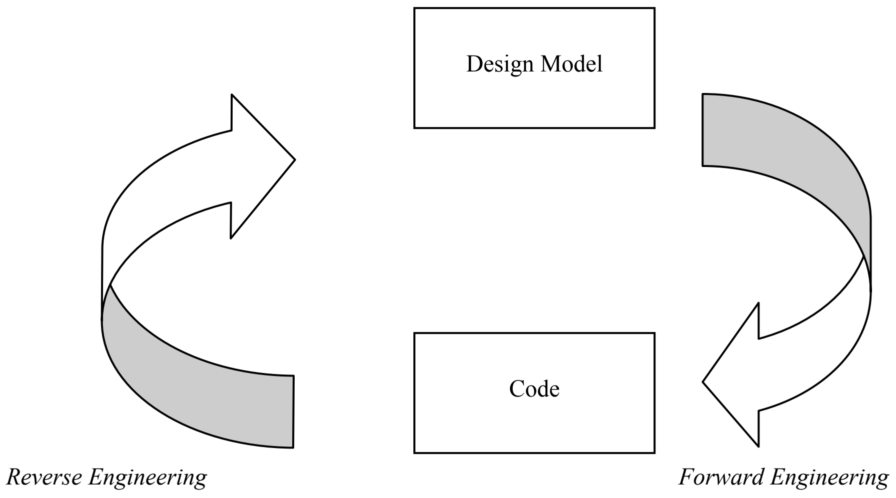

Figure 10.3: Reverse Engineering and Forward Engineering

Check your progress 2

1. After Re-engineering, the new system becomes ___________ and ___________.
2. __________ is the process of generation of skeleton code out of the design models.
3. All case tools are based on __________, which is particularly useful when the user requirements are difficult to define.

10.4 VISUAL AND EMERGING CASE TOOLS

Visual CASE tools enable user to quickly create user interface and related skeleton code.

10.4.1 Traditional systems development and CASE based systems development

The following are the features of Traditional systems development:

- Emphasis on program coding, testing and documentation
- Specifications are to be written by the analyst and are paper-based
- Coding is manual and tedious
- Documentation is written by the programmer or the analyst and are often done after completion of the process
- Software testing process follows the traditional approach
- Manual maintenance of code and documentation.

The following are the features of CASE-based systems development:

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

10.4.2 CASE environment

The earlier generation of CASE tool developers concentrated to a large extent on the automation of isolated tasks of system development process, such as document production, version control of source code, and design method support. Need of integrating these tools to support a common development environment was felt to support the activity effectively. 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. Figure 10.4 depicts typical CASE environment.

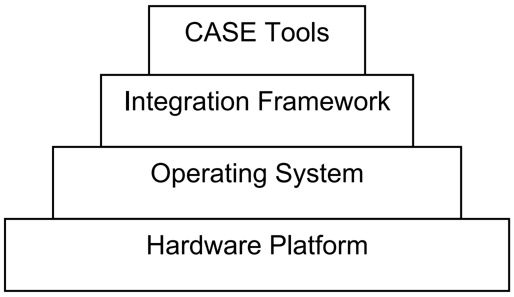

Figure 10.4 : CASE environment

The key to CASE tools is integration. Tools are more effective if they are integrated to work together. Integration could be data integration, user interface integration or activity integration.

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.

Rational Rose is one of the most widely used CASE tools by the software community. The teams responsible for software development are finding that modeling using CASE tools are becoming increasingly important for the software development process. Software developed using model driven technique such as Rational Rose offer a range of products to suit different activities of software development process.

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

10.4.3 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 gamut of 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.

Check Your Progress 3

1. Oracle Designer 2000 is a _________ CASE tool
2. Enterprise Architect is an object oriented CASE tool which can be used during ______ phases of life cycle
3. All integrated CASE tools usually share _________ user interface

10.4.4 Object Oriented 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).*

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. Our discussion here will describe object oriented CASE tools in general without making references to a specific product.

The following are some of the features found across most of the Object Oriented CASE tools:

1. Create graphics, such as class diagrams, message diagrams, state diagrams etc.
2. Create text specifications such as system specification, class specification and relationship specification.
3. Generate source code.
4. Repository of models.

Differences between various types of object oriented CASE tools

All of these object oriented CASE tools are similar in terms of their capabilities to create class diagrams, code generation. However, they may differ in terms of their extendibility, number of supported operating systems and additional features and capabilities, such as support for different methodologies and computer language code generation (C++, Ada, Java etc). Most provide capability to automatically generate code of C++ and other languages from a class diagram and class specifications. Some provide the capability to generate class diagrams from code (reverse engineering). Figure 10.5 depicts drawing and text tools.

An object oriented CASE tool has the capabilities of drawing class diagrams and state diagrams. Specification tools create text reports.

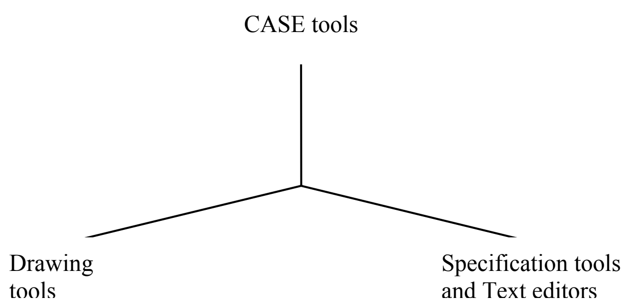

Figure 10.5: Drawing and text tools

Figure 10.6 depicts major outputs of object oriented CASE tools.

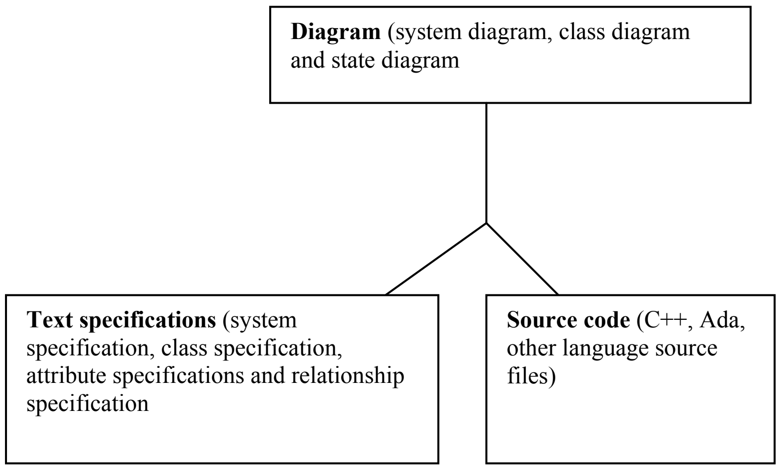
Diagram (system diagram, class diagram and state diagram
Text specifications (system specification, class specification, attribute specifications and relationship specification
Source code (C++, Ada, other language source files)

Figure 10.6: Major Outputs of Object Oriented CASE Tools

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.

Figure 10.7 depicts the class diagram for a typical Banking application.

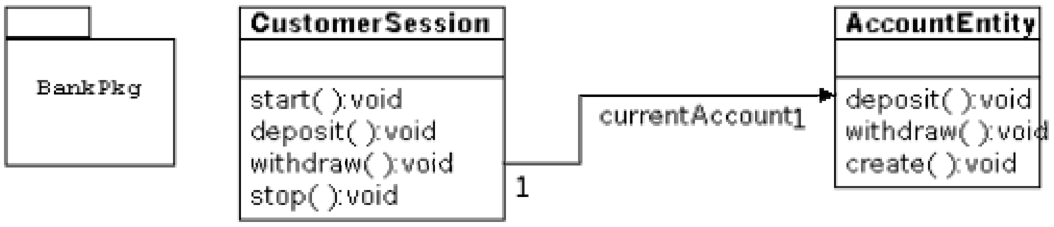

Figure 10.7 : A class Diagram for a typical Banking Application

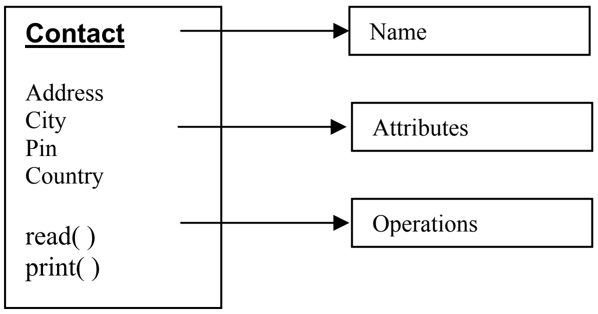

Figure 10.8: An Example of a Simple Class *Contact*

10.4.5 Creating documentation and reports using object oriented 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.

Most of 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. 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. Figure 10.9 depicts the process to create a model document.

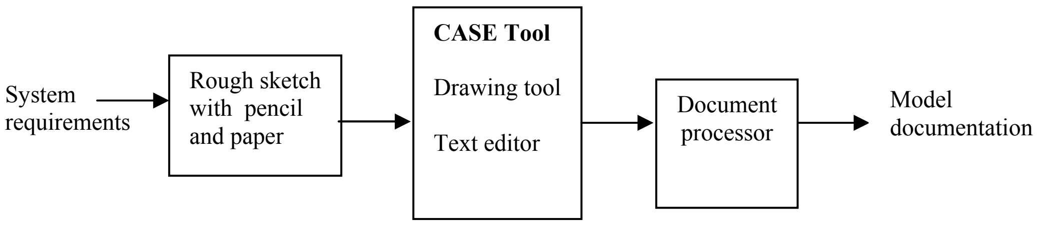

Figure 10.9 : Process to create a model document

10.4.6 Creating an executable prototype using object oriented CASE tools

Most of the object oriented CASE tools greatly assist in creating executable prototypes based on design specifications. The diagram below shows how CASE tools are used to create executable codes. System requirements are prepared in text and diagrams by pencil and paper. Design specifications like class specifications, system diagrams and various text specifications are then generated using tools available in CASE. From this, design specification codes are generated using code generation tools. Most of the CASE tools support generation of C++ code whereas some may support other languages as well. The source code generated might require updation with formulas, expression, etc. CASE tools also create updated diagrams based on updated source code. Figure 10.10 depicts the process to create an executable prototype using a CASE tool.

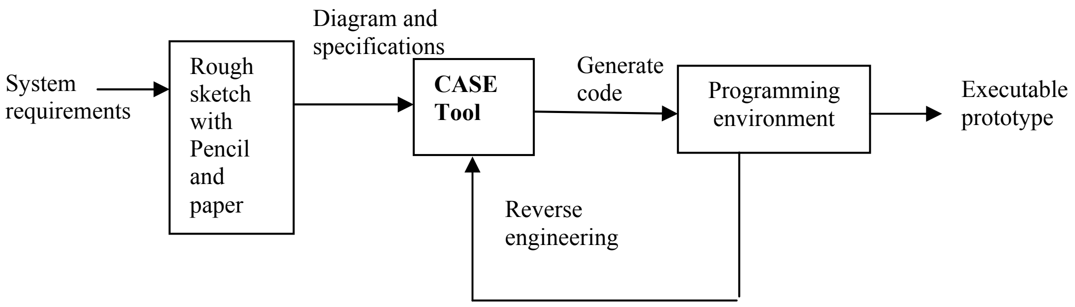

Figure 10.10 : Process to create an executable prototype using a CASE tool

10.4.7 Sequence Diagrams

UML provides a graphical means of depicting object interactions over time in Sequence Diagrams. These typically show a user or actor, the objects and components they interact with in the execution of a use case. One sequence diagram typically represents a single Use Case scenario or flow of events.

Sequence diagrams are an excellent way to document usage scenarios and to both capture required objects early in analysis and to verify object usage later in design. Sequence diagrams show the flow of messages from one object to another, and as such correspond to the methods and events supported by a class/object.

Figure 10.11 shows an example of a sequence diagram, with the user or actor on the left initiating a flow of events and messages that correspond to the Use Case scenario. The messages that pass between objects will become class operations in the final model.

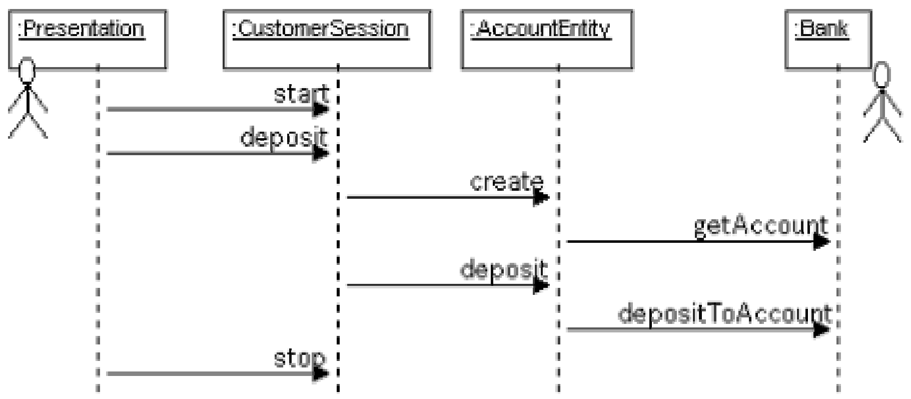

Figure 10.11: A typical Sequence Diagram for deposit into Account

Check Your Progress 4

1. The process of generating class diagram from code is called ______
2. _______________ are an excellent way to document usage scenarios and to both capture required objects early in analysis and to verify object usage later in design.

10.5 SUMMARY

Changes are occurring in traditional way of system development. Organizations use CASE tools to improve integration and development activities. The tools also help standardize the development process. Computer Aided Software Engineering (CASE) automates part of system development process. 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.

10.6 SOLUTIONS/ ANSWERS

Check Your Progress 1

1. CASE tools
2. Speed, quality
3. Automation

Check Your Progress 2

1. Restructured, Redocumented
2. Forward Engineering
3. Prototyping

Check Your Progress 3

1. Visual
2. All
3. Common

Check Your Progress 4

1. Reverse engineering
2. Sequence diagrams

10.7 FURTHER READINGS

Joey George, J. Hoffer and Joseph Valacich; *Modern Systems Analysis and Design*, *Third Edition, 2001*, Pearson Education.

James A. O'Brien; *Introduction to Information Systems*, *An End user/Enterprise Perspective*; Mc Graw Hill Edition; 1995

Reference Websites

http://www.sei.cmu.edu/legacy/case/case_whatis.html
http://www.rational.com/product