BACHELOR OF COMPUTER APPLICATIONS

SEMESTER 6

DCA3245

SOFTWARE PROJECT MANAGEMENT

Unit 11

Computer Aided Software Engineering (CASE) Tools

Table of Contents

1. INTRODUCTION

In the previous unit, we have discussed the importance of quality in software development. In this unit, let's discuss various standard tools available for software development.

In today's world the focus is on speed. It is applied to every aspect of life and especially to the technological scene. Because of the ever changing market dynamics and advent of newer technology the products are being replaced by their newer versions ever so faster nowadays. And most of these products are software driven right from automobile field to home entertainment system. Hence the software that support these new products with its new feature and design also should be developed that much faster. This puts in to a totally new perspective on software development. Software development in earlier days used traditional approach where each phase or step is carried out after the completion of preceding step. However this traditional approach is very time consuming and is not very flexible in terms of managing changes to the software. Speeding up the development time is not very easy as it is hard to find the bottleneck in the development process. Hence, the new paradigm of software development was introduced around 70's to speed up the development activities using computer programs itself. This also helped by the fact that the computing speed is also increased tremendously over these years. This concept of software development using the supporting software is termed as Computer Aided Software Engineering (CASE) we will be discussing CASE in detail in this unit.

1.1 Objectives:

After studying this unit, you should be able to:

- *Explain the basic concepts of CASE*
- *Discuss how CASE supports software life cycle*
- *Classify CASE tools*
- *Describe integrated CASE environments*
- *Explain the architecture of CASE environment*
- *Discuss CASE repository*

2. CASE CONCEPTS

The term CASE is applied to software products development that uses extensive software engineering principles and these processes are implemented either partly or majorly through the supporting software. In other words, the software development is accomplished taking help of another software as development tool. In a way we can consider this as online programming. It is not that easy to implement such a software development activity as it can be very complex and hard in terms of maintainability of the application. As the development tools themselves will be progressive also, when the tools become obsolete with newer version coming in the market, the software developed such CASE tools at risk of being out of date from maintenance point of view.

The tools that help software development were first conceived around 1970's. The earlier versions of the tools were developed with the intention of automating the production line and also help with structured drawings. If the entire software development life cycle is covered by these automation tools, such environment is known as *Integrated Computer Aided Software Engineering (I-CASE).* If only certain processes are covered by the automation tools, then it is known as commonly known CASE, the just *Computer Aided Software Engineering (CASE).* The initial developments in the field of automation tools were focused on CASE, later the I-CASE tools were introduced to the market. The I-CASE tools are capable of generating the entire applications right from the design specifications.

The third generation of CASE tools has come to the market now, with the intention of further speeding up the product development cycle. It includes improvements to capabilities of I-CASE tools and also incorporates the new methodologies. These new methodologies are mainly based on the rapid prototyping. Prototyping is sample model or release of a product built, to test the concepts. The prototype can be developed in an iterative fashion adding features incrementally and testing the product in the process. This rapid prototyping has the advantage that the product can be developed faster and tested more often between the development phases catching the errors and defects in the early stage of development. As we all know catching the defect and fixing it is much cost effective when compared to the defect being identified and re-worked at final stages of the project.

CASE is rapidly growing technology; new set of tools are being developed based on business needs and strategic operational needs. At the very fundamental level, the I-CASE tool consists of a repository. This repository acts as the knowledge base for the organization capturing the information about the organization, its organizational structure, functioning details, data models, procedural details etc. Further the repository consists of details like diagrammatical design, the design methodology as well. The repository grows over the years collecting the details of the work done and acting as the rich source of information that can be retrieved when needed by the organization*. Hence this repository is known as the heart of the CASE system.*

Two base mechanisms of CASE software to store the information are:

1) A *dictionary*, which contains names and descriptions of data items, processes, etc.
2) A *repository*, which contains this dictionary information and a complete coded representation of plans, models and designs, with tools for cross-checking,

CASE TOOL:

As the 1990s progressed, the word "CASE tool" entered the software vernacular, and by the end of the decade, large companies like IBM were using them to streamline the software development process. The term "CASE" is used to describe a wide range of tools that aid in the software development life cycle at various times. Code changes can be managed more easily with the help of version control software. These resources facilitate teamwork, version control, and the seamless integration of new code from a large number of developers. This kind of software is useful for keeping track of a project's tasks, due dates, resources, and general development. They help keep the project on track by keeping an eye on its progress.

Components of CASE tool: Distinct parts of a CASE tool are devoted to distinct stages of the software development life cycle. CASE tools provide a faultless framework for entire software systems, beginning with the initial project plan and continuing all the way through post-implementation maintenance, design, and testing.

1. Upper Case Tool: Because they are the foundation of the systems being developed, upper case tools are the backbone of software development efforts. They improve the software development process in various ways, including planning, analysis, and design.
2. Lower Case Tool: The lowercase tools are the next step in the process after the uppercase ones. They are linked to the phases of software development projects that involve putting the code into action, testing it, and keeping it running.
3. Integrated software tools: From initial data collection to final documentation and testing, they are employed at every stage of the software development life cycle. Managing tasks and preserving design documents for the duration of a system's development life cycle. The data for all of these tools and system development operations is stored in a central repository. They are crucial at every level of the software development process, from brainstorming to testing to documentation.

CASE Tool Repository:

The consolidated database is the nerve centre for all project management data. Product specs, reports, diagrams, documents, and other helpful data pertaining to management as a whole are all housed here. It's been called the "data hub" of software development.

CASE classification: There are a number of distinct types of CASE tools that serve different purposes and are used at different points in the software development life cycle. CASE tools can be broken down into the following categories. Based on their principal purposes and the software development life cycle phases they facilitate, CASE tools fall into a number of distinct types.

- Tools for Making Flowcharts and Other Diagrams Flowcharts, UML diagrams, data models, and entity-relationship diagrams are just a few examples of the visual representations of software architecture that may be generated with the assistance of these technologies.
- Analysis Instruments for Requirements Devices in this class facilitate the collection, organisation, and evaluation of requirements from various parties.
- Requirements specification, traceability, and management tools may be included.

- Methods of Design: High-level and granular design specifications can be made with the use of these tools. Software for creating everything from buildings to databases to user interfaces could fall under this category.

1. Language centered are the Tools for Automatic Code Generation and Programming that is Code can be generated from these high-level design specifications with the help of these tools. Integrated development environments (IDEs) for creating and modifying code may also be included.
   - Methods of Testing: Unit testing, integration testing, and system testing can all go smoothly with the help of these instruments. Frameworks for managing test cases and automated testing tools are two examples.
   - Tools for Debugging: Debugging tools aid in finding and fixing software flaws. Features like stepping through code, inspecting variables, and viewing the call stack are all available.
2. Structured centered based on the concept of environment generation:
   - Software for Documentation: User guides, technical specifications, and system documentation are just few of the types of writing that can be generated automatically by using such a technology.
   - Software for Managing and Scheduling Projects: These applications are useful for keeping track of projects and their tasks, as well as for planning and scheduling them. They aid in keeping the project on track and under control.
   - Software for Managing and Maintaining Versions and Configurations: Using these tools, you can be sure that the right code versions are being utilised and that dependencies are being managed

The CASE tools can be broadly categorized as follows:

1) *Information engineering-supporting products.* This class of tools support the software project life-cycle processes. These processes are derived from strategic plans of the organization and provide a repository to set up and maintain the enterprise models, data models, and process models.

2) *Structured diagramming-supporting products.* These products at least support data flow, control flow and entity flow, which are the three basic structured software engineering diagram types.
3) *Structured development aids-providing products.* These products are providing aids for a structured development of the process. These products are very suitable to be used by the system analysts, because they are helped very much by a structured process, because those can be analyzed faster and more accurately.
4) *Application-code-generating products:* These are products that generate application-code for a specific goal, set by the designer. The code generation happens based on the target code required for the organization and corresponding tool has to be selected for the code generation purpose. Tools will be available to generate the code in popular programming languages like COBOL, Java, or C++ etc.

Computer Aided Software Engineering (CASE) tools are of great help to software project managers, and team members who work on software development related activities. The tools automate the common project follow up tasks (like schedule check, or deviation determination etc.) and help developers and tech leads in their analysis, design, coding, and testing work.

Computer Aided **Software Engineering (CASE)** is the use of software tools to assist in the development and maintenance of software. Tools used to assist in this way are known as CASE Tools

The Computer Aided Software Engineering methodology helps the software engineer with automation activities of routine work. This in turn provides the software developer insight to engineering aspects and helps concentrate on the problem analysis. Also the CASE tools will have the pre-defined quality standards to be met that are defined during the inception of the tool idea, Some typical CASE tools are:

- Code generation tools
- Data modeling tools

- UML modeling tools
- Code and Design Refactoring tools
- QVT or Model transformation Tools
- Configuration management tools including revision control

The CASE software tools can support all the phases and processes of the software life cycle. Hence the use of the CASE tools can be found right from project management to the activities of software product development like requirement gathering, design, code generation, testing, and also tasks like compilers. Although the CASE tools are in widespread use for all the project aspects, the most widely used tools are for analysis and design activities. Further we have to be very careful about the categorization of CASE tools used for a project and tool usage itself. The CASE tools must be applied after due diligence on the parts that would help the project if automated using a CASE tool. Otherwise the project execution might suffer if the tools that are not appropriate for current project are forcibly used upon in the project.

SELF-ASSESSMENT QUESTIONS - 1

1. The term CASE is applied to software products development that uses extensive software engineering principles and these processes are implemented either partly or majorly through the supporting software. (True / False)
2. CASE stands for _____.
3. The _____ is the heart of a CASE system. (Pick right option)
   a) Repository
   b) Dictionary
   c) Design
   d) Coding

3. CLASSIFICATION OF CASE TOOLS

The CASE tools are classified based on the functionality that they perform, by the role the tools play in the context of the project when the tools are used by the managers and the practitioners, by their use in various phases of the project, and also by the environment that is needed to support the tools in terms of hardware and software support. In some scenarios the classification can even be based on the cost origin of the tools. Some common tool classification is presented here.

Business Process Engineering Tools: The primary focus of these tools is to model the business information by modeling the key details of the organization, and the requirements of the organization. By doing so Business Process Engineering tools aims for representing the business processes and their interactions. This tool provides the basic model of the information which acts as the meta-data from which the specific information can be derived. It encompasses and models the business entities of the organization and their interactions rather than focusing on specific requirements of any particular business of the organization.

Process Modeling and Management Tools: If an organization wants to improve upon its business or software processes, it must first understand the processes to be improved. In order to ensure that the entire team is aware of the current state and the state of processes organization trying to target for, these process modeling and management tools can be used. These tools model the processes by representing the key elements of the process. With the help of the model the team can understand the change that management is trying to bring in and participate in the change with full understanding.

Project Planning Tools: These tools focus on two main types of tools: software project effort and cost estimation tools and project scheduling tools. Estimation tool used estimates the in terms of effort, duration, and number of resources needed for the project execution. On the other hand the Project scheduling tools help the project manager to list out all the tasks and arrange them based on the dependencies among the tasks. Scheduling tools help the manager to also track the schedule as the project progresses.

Risk Analysis Tools: As we have seen from previous discussions, the risk management is very critical for the success of the project. Especially for the large project, project manager must ensure that risk plan and risk mitigation plans are maintained and followed upon. Risk analysis tools help project manager in risk assessment activities and also in related analysis.

Project Management Tools: The project schedule and project plan must be tracked and monitored on a continuous basis. In addition, a manager should use tools to collect metrics that will ultimately provide an indication of software product quality.

The following tools are extensions to project planning tools.

Requirements Tracing Tools: When large systems are developed, things "fall into the cracks." That is, the delivered system does not fully meet customer specified requirements. The objective of requirements tracing tools is to provide a systematic approach to the isolation of requirements, beginning with the customer request for proposal or specification. The typical requirements tracing tool combines human interactive text evaluation with a database management system that stores and categorizes each system requirement that is "parsed" from the original RFP or specification.

Metrics and Management Tools: Software metrics is very important entity for the improvements of the project. It is the statistical analysis of the proceedings of the project. Metrics provide the manager ability to deal with process control and coordination, while the other team members like developers, testers can use the metrics to improve on the quality of their deliverables. Some of the commonly captured metrics by the tools are LOC (line of code) per person-hour, defect density, defect per function point etc.

Documentation Tools: Document production and desktop publishing tools support nearly every aspect of software engineering and represent a substantial "leverage" opportunity for all software developers. Most software development organizations spend a substantial amount of time developing documents, and in many cases the documentation process itself is quite inefficient. It is not unusual for a software development organization to spend as much as 20 or 30 percent of all software development effort on documentation. For this reason, documentation tools provide an important opportunity to improve productivity.

System Software Tools: CASE is a workstation technology. Therefore, the CASE environment must accommodate high-quality network system software, object management services, distributed component support, electronic mail, bulletin boards and other communication capabilities.

Quality Assurance Tools: The majority of CASE tools that claim to focus on quality assurance are actually metrics tools that audit source code to determine compliance with language standards. Other tools extract technical metrics and in an effort to project the quality of the software that is being built.

Database Management Tools: Database management software serves as a foundation for the establishment of a CASE database (repository) that we have called the *project database*. Given the emphasis on configuration objects, database management tools for CASE are evolving from relational database management systems to object-oriented database management systems.

Software Configuration Management Tools: SCM plays very important role in the success of the project. Tools can assist in all five major SCM tasks – identification, version control, change control, auditing, and status accounting. The CASE database or repository provides the mechanism for identifying the configuration items, change control process, running the audit, and status accounting (through communication tools).

Analysis and Design Tools: Analysis and design tools enable a software engineer to create models of the system to be built. The models contain a representation of data, function, and behavior (at the analysis level) and characterizations of data, architectural, component-level, and interface design. By performing consistency and validity checking on the models, analysis and design tools provide a software engineer with some degree of insight into the analysis representation and help to eliminate errors before they propagate into the design, or worse, into implementation itself.

PRO/SIM Tools: PRO/SIM (prototyping and simulation) tools provide the software engineer with the ability to predict the behavior of a real-time system prior to the time that it is built. In addition, these tools enable the software engineer to develop mock-ups of the real-time

system, allowing the customer to gain insight into the function, operation and response prior to actual implementation.

Interface Design and Development Tools: Interface design and development tools are actually a tool kit of software components (classes) such as menus, buttons, window structures, icons, scrolling mechanisms, device drivers, and so forth. However, these tool kits are being replaced by interface prototyping tools that enable rapid onscreen creation of sophisticated user interfaces that conform to the interfacing standard that has been adopted for the software.

Prototyping Tools: A variety of different prototyping tools can be used. *Screen painters* enable a software engineer to define screen layout rapidly for interactive applications. More sophisticated CASE prototyping tools enable the creation of a data design, coupled with both screen and report layouts. Many analysis and design tools have extensions that provide a prototyping option. PRO/SIM tools generate skeleton Ada and C source code for engineering (real-time) applications. Finally, a variety of fourth generation tools have prototyping features.

Programming Tools: The programming tools category encompasses the compilers, editors, and debuggers that are available to support most conventional programming languages. In addition, object-oriented programming environments, fourth generation languages, graphical programming environments, application generators, and database query languages also reside within this category.

Web Development Tools: The common activities involved in developing a web based application are automated in this type of tools. The activities automated include the meta-information creation, creating the archive file containing the text, graphics, and other types of web pages, maintain the appropriate relation in web definition file (web.xml) etc.

Project Management : These tools help in planning of the project, estimation of cost and efforts, scheduling of project and planning of resources. All the steps in the execution of the project must be strictly followed by the managers in management of software project. The tools of project management helps in storing and sharing the project information in real time using throughout the organization. Planning, organising, executing, and controlling the

resources and actions necessary to achieve certain goals within a set time frame constitutes the core activities of project management. The term "project management" refers to the process of overseeing a project from its inception until its completion. Timely, cost-effective, and satisfactory project completion is the result of competent project management.

Documentation Tool: With the help of these technologies, it is possible to automatically generate user guides, technical specifications, and other project documents. The end user documentation details the system's operation and functionality. The end user documentation explains the system's inner workings and how it operates.

It's important to start documenting the software project before you even start writing code.All stages of the software development life cycle, including the finalisation of the software development phase, contribute to the documentation. The software can provide documentation for both developers and regular consumers. Use Doxygen, Adobe RoboHelp, DrExplain, etc., as an illustration.

With the right software, you can easily make and update architecture diagrams, flowcharts, class diagrams, and other types of UML diagrams.Code documentation generation capabilities are included in many Case Tools. One use case is the generation of API documentation from inline comments automatically. You may construct thorough test plans and scripts with the help of a Case Tool, which you can then use to create and manage test cases.The documentation for a given build of the programme can be linked to that build specifically through integration with version control systems.

Analysis Tool:

Equipment in this category helps with gathering, organising, and assessing requirements from several sources. Tools for requirements management, specification, and traceability may be provided. These instruments are useful for gathering specifications and checking visually for mistakes or omissions in data or diagrams.

These aids Finds errors in charts, spreadsheets, and reports. Using the transparent analyst, any analysis may be performed. Analysing information, operations, and structures requires the use of dedicated applications called "analysis tools." Many different industries and

disciplines use these techniques to analyse data, identify trends, and arrive at sound conclusions. These examples highlight the breadth of fields and applications that can profit from the many specific analytical tools currently available. The analysis at hand, the information at hand, and the specific requirements of the user or group all play a role in determining the optimal tool.

SELF-ASSESSMENT QUESTIONS - 2

4. Business process engineering tools provide the basic model of the information which acts as the meta-data from which the specific information can be derived. (True / False)
5. _____ model the processes by representing the key elements of the process.
6. Screen painter is an example for _____ tools. (Pick right option)
   a) Programming
   b) Prototyping
   c) Design
   d) Analysis

4. STEPS FOR CASE TOOL IMPLEMENTATION

Now let's see various steps for implementing CASE tools.

1) Conduct a technology-impact study to determine how the basic business of the organization should change to maximize the opportunities presented by rapid technological change.
2) Evaluate how the organization should be re-engineered to take advantage of new technology.
3) Establish a program for replacing the old systems with the most effective new technology.
4) Commit to an overall integrated architecture.
5) Select a development methodology.
6) Select a CASE tool.
7) Establish a culture of reusability.
8) Strive for an environment of open interconnectivity and software portability across the entire enterprise.
9) Establish inter-corporate network links to most trading partners.
10) Determine how to provide all knowledge to workers with a high level of computerized knowledge and processing power.
11) Determine the changes in management-structure required to take full advantage of innovative systems, architectures, methodologies and tools.

5. INTEGRATED CASE ENVIRONMENTS

An I-CASE environment is an integrated approach; it is a combination of pool of CASE tools and other components that supports most or all of the integration that occurs among the environment components, and between users of the environment and environment itself. From the definition, the core part is that the interactions among environment components are supported within in the environment. The discrimination of CASE environment from a random combination of CASE tools is there is a physical mechanism (shared database or message broadcast), provided in the environment that facilitates interaction of those tools.

The integrated CASE (I-CASE) include (1) smooth transfer of information (models, programs, documents, data) from one tool to another and one software engineering step to the next; (2) a reduction in the effort required to perform umbrella activities such as software configuration management, Quality assurance, and document production; (3) an increase in project control. That is achieved through better planning, monitoring, and communication; and (4) Improved coordination among staff members who are working on large software project. But I-CASE also poses significant challenges. Integration demands consistent representations of software engineering information, standardized interfaces between tools, a homogeneous mechanism for communication between the software engineer and each tool, and an effective approach that will enable I-CASE to move among various hardware platforms and operating systems. Comprehensive I-CASE environments have emerged more slowly than originally expected. However, integrated environments do exist and are becoming more powerful as the years pass.

The term *integration* implies both *combination* and *closure*. I-CASE combines a variety of different tools and a spectrum of information in a way that enables closure of communication among tools, between people, and across the software process. Tools are integrated so that software engineering information is available to each tool that needs it; usage is integrated so that a common look and feel is provided for all tools; a development philosophy is integrated, implying a standardized software engineering approach that applies modern practice and proven methods.

To define integration in the context of the software engineering process, it is necessary to establish a set of requirements for I-CASE: An integrated CASE environment should provide the following,

- Provide a mechanism for sharing software engineering information among all tools contained in the environment.
- Enable a change to one item of information to be tracked to other related information items.
- Provide version control and overall configuration management for all software engineering information.
- Allow direct, non-sequential access to any tool contained in the environment.
- Establish automated support for the software process model that has been chosen, integrating CASE tools and software configuration items (SCIs) into a standard work breakdown structure.
- Enable the users of each tool to experience a consistent look and feel at the human/computer interface.
- Support communication among software engineers.
- Collect both management and technical metrics that can be used to improve the process and the product.

There are many ways to form CASE environment, one possible way of providing the 'glue' that links CASE tools together, it definitely leads to a spectrum approaches to implementing a CASE environment.

The first step to form CASE environment is selection of the case tools and components. The second step is assembling case environment by considering the interaction among the components. Virtually many people concentrating on first step, they are ignoring interaction among the components. But we have to concentrate much more on how to select components and made them to work together effetely, and concentrate less on selection of components. The chosen approach of interaction of components in the given context will depend on many overlapping factors like the availability of resources, the need of organization in question

and so on. So it is necessary to determine the related factors and constrains for CASE environment most suited to the problem at hand.

SELF-ASSESSMENT QUESTIONS – 3

7. The term integration implies both combination and closure. (True / False)
8. SCI stands for _____.

6. ARCHITECTURE OF CASE ENVIRONMENT

Architecture refers to the way a system is designed and how the components are connected with each other. There are computer architectures, network architectures and software architectures.

IT architecture is a design for the arrangement and interoperation of technical components that together provide an organization its information and communication infrastructure.

Architecture is the basis for building any software or hardware systems. Here we are going to discuss the ICASE architecture. A software engineering team uses CASE tools, corresponding methods, and a process framework to create a pool of software engineering information. The integration framework facilitates transfer of information into and out of the pool. To accomplish this, the following architectural components must exist: a database must be created (to store the information); an object management system must be built (to manage changes to the information); a tools control mechanism must be constructed (to coordinate the use of CASE tools); a user interface must provide a consistent pathway between actions made by the user and the tools contained in the environment. Most models of the integration framework represent these components as layers. A simple model of the framework, depicting only the components just noted is shown in figure 11.1.

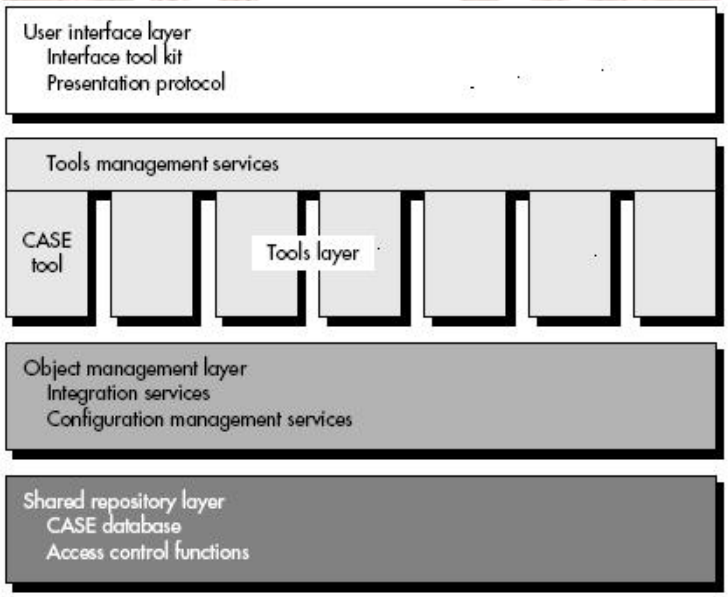

Fig. 11.1: Integrated CASE environment

The *user interface layer* (Figure 11.1) incorporates a standardized interface tool kit with a common presentation protocol. The interface tool kit contains software for human/computer interface management and a library of display objects. Both provide a consistent mechanism for communication between the interface and individual CASE tools. The *presentation protocol* is the set of guidelines that gives all CASE tools the same look and feel. Screen layout conventions, menu names and organization, icons, object names, the use of the keyboard and mouse, and the mechanism for tools access are all defined as part of the presentation protocol.

The *tools layer* incorporates a set of tools management services with the CASE tools themselves. *Tools management services* (TMS) control the behavior of tools within the environment. If multitasking is used during the execution of one or more tools, TMS performs multitask synchronization and communication, coordinates the flow of information from the repository and object management system into the tools, accomplishes security and auditing functions, and collects metrics on tool usage. In essence, software in this layer of the framework architecture provides the mechanism for tools integration. Every CASE tool is "plugged into" the object management layer. Working in conjunction with the CASE repository, the Object Management Layer (OML) provides integration services – a set of standard modules that couple tools with the repository. In addition, the OML provides configuration management services by enabling the identification of all configuration objects, performing version control, and providing support for change control, audits, and status accounting. The *shared repository layer* is the CASE database and the access control functions that enable the object management layer to interact with the database.

CASE Repository

CASE Repository is a database that acts as the center for both accumulation and storage of software engineering information. The role of the person (the software engineer) is to interact with the repository using CASE tools that are integrated with it. The repository is a relational or object-oriented database that is "the center of accumulation and storage" for software engineering.

Assistant Modules

The instruments themselves are instruments. These carry out a certain duty within a given stage of growth, either automatically or with some human intervention. Assisting tools perform quality assurance checks on the requirements models and propagate any modifications made to one model to the others during this phase.

If the user modifies some verbal descriptions, the related graphical models would need to be redrawn. The field of human-computer interaction has been rapidly developing during the past few decades.

The human-computer interaction (HCI) technology used by CASE tools has steadily progressed from the teletype-based interactions of the past to the GUI-based interactions of today's multitasking environments.

Communication Component: Data sharing between CASE applications is handled through the communications module. There are a number of compelling arguments for resolving the issue of poor inter-CASE tool communication. Due to a lack of industry-wide standards, CASE tool makers have implemented a broad variety of proprietary, frequently incompatible formats for storing development data within their applications.

These apps streamline group communication, collaboration, and the distribution of finished items. Communications of various kinds, including SMS, comments, notifications, etc. These tools make it easier to gather, organise, and assess needs from a wide range of stakeholders. One example is software designed to help with managing and keeping track of needs.

If the user modifies some verbal descriptions, the related graphical models would need to be redrawn. The field of human-computer interaction has been rapidly developing during the past few decades.

The human-computer interaction (HCI) technology used by CASE tools has steadily progressed from the teletype-based interactions of the past to the GUI-based interactions of today's multitasking environments.

Process Modelling

The software process model can be created using these modelling tools for software development. The managers can choose a process model or make modifications depending upon the requirements of the software product

- The creation of visual representations of the software development process model is greatly aided by modelling tools, particularly those that offer flowcharting or process diagramming. Processes like requirements analysis, design, development, testing, and rollout are all a part of this.
- These software development modelling tools can be used to build the software process model. Managers are given the freedom to select a process model or make adjustments based on the needs of the application. Managers may make better decisions about the software development process, one that is personalized to the project's unique requirements and goals, when they use modelling tools in this way. As a result, this improves the overall quality and success of the software.
- Example for process modelling is EPF composer , or Eclipse PfC, is a programme created by the Eclipse Foundation to help in the composition of Eclipse-based process frameworks. You can use it to design and share your own process models. The Eclipse Process Framework (EPF) project, of which EPF Composer is a component, strives to supply a modular framework for specifying and carrying out software development procedures.

Advantage of CASE Tools:

- The three Cs: CASE tools provide software project consistency, completeness, and conformity under optimal conditions.Quickness and efficiency:
- Software development projects can save time and money by making use of a wide range of CASE tools.
- Effective operational life is increased and real-world user needs are more likely to be met with the use of CASE tools, which facilitate a more streamlined development process. Spend less on repairs thanks to careful planning and testing during the designing phase.

- CASE tools offer a unified setting for the creation of intricate projects.
- The total cost of maintenance over the product's lifetime is cut significantly when more effort is put into its design and testing phases.
- Implementing a systematic strategy during development improves the final result.
- The use of computers in the software development process improves the likelihood of satisfying practical needs.

CASE environment VS Programming Environment:

Thanks to CASE tools, the effort and time spent on software upkeep have been cut significantly. This is because tracking and ensuring consistency are far easier in a CASE setting.

When utilised in a CASE context, which facilitates the systematic collecting of data across the many stages of software development, tools like these become far more helpful and efficient.

An organization's culture can change for the better when a CASE environment is put into place. For example, software developers can instantly and automatically balance DFDs at the touch of a button.

Striking Features:

- Many routine tasks in the software development process are now automated thanks to CASE technologies. This allows programmers to devote more effort to intricate design and issue resolution.
- Reduced Cycle Times in a Hurry: Time to market can be reduced and improved by using CASE technologies to automate tasks like code generation, documentation creation, and testing.
- CASE tools often include features for code analysis, debugging, and testing, which helps catch and repair issues before they spread. This leads to better finished products in terms of software.
- Information such as project plans, specifications, and source code can be stored in one convenient spot with the help of CASE solutions. Better teamwork and fewer misunderstandings are the end consequence.

- Standardisation of Procedures: CASE tools make guarantee that everyone on the team adopts the same processes and coding standards. The resulting code is more stable and simpler to update.

Implementing CASE Tools:

- In the software development process, CASE technologies automate numerous previously manual and repetitive operations. This frees up developers' time for more complex design and problem solving.
- Rapid Cycle Time Improvements: Automating processes like code generation, documentation creation, and testing with the help of CASE tools shortens development times and increases time to market.
- Superior Quality: Features for code analysis, debugging, and testing are common in CASE tools, allowing for problems to be fixed at an early stage of development. As a result, the quality of software produced improves.
- CASE solutions provide a centralised location for team members to store and easily access all relevant project information, including drawings, documentation, and code. The result is better team cohesion and fewer misunderstandings.
- Procedures Standardisation: CASE tools ensure that everyone in the team follows the same procedures and uses the same coding conventions. This results in more consistent and easier to maintain code.
- The Software Development Life Cycle should be a central part of the approach behind CASE tools. CASE tools need to be compatible with the organization's current development environments.
- Create Industry Standards: Establish and disseminate guidelines for effective CASE tool utilisation. Standards for code, documentation, and version control, among other practises, may be included here. Sharing Information and Documenting Processes Give people all they need to know about using CASE to execute their jobs well. Inspire team members to work together and share what they've learned. Assuring and Testing Quality will Test the CASE tools extensively to make sure they perform as advertised and don't cause any problems during development. Implementation and Widespread

Use as If the pilot is a success and any tweaks are made, the CASE tools can be introduced to the rest of the development team.

- Control and Assistance: Maintain an eye on how the CASE tools are being put to use and always be there to help out the dev crew. Take care of any problems that arise, offer more training if required, and make sure the resources are still useful to the team for Analysing and bettering performance:

An efficient CASE tool must possess the following characteristics :

- A standard methodology: A CASE tool must support a standard software development methodology and standard modelling techniques.
- Flexibility: Editors and other tools must be flexible in usage.
- The CASE tool must offer flexibility and the choice for the user of editors' development environments.
- A CASE tool's viability depends on its ability to support a standardised software development methodology and modelling practises. Editors and other tools need to be adaptable to different user needs. The CASE tool should allow the user to select among multiple editing and development contexts.

CASE software Development Environment:

CASE tools support extensive activities during the software development process.The functional features provided by CASE tools for the software development process are:

- Creating software requirements specifications
- Creation of design specifications
- Creation of cross references.
- Verifying/Analysing the relationship between requirement and design
- Present CASE tools support Unified Model Language (UML).
- Performing project and configuration management
- Building system prototypes
- Containing code and accompanying documents.
- Validation and verification, interfacing with external environment

With a CASE (Computer-Aided Software Engineering) software development environment, developers have access to a wide variety of resources for every stage of the software creation process. It is a collection of automated tools and resources for creating, testing, and managing software systems that are used by developers and teams.

Requirement of a prototyping CASE Tool:

A prototype is an early model or version of a significant component of a system or a miniature representation of the complete system. To satisfy the needs of the development process, the right CASE (Computer-Aided Software Engineering) tools for prototyping must be selected and put into place.

Prototyping helps with many different things, like showing a concept, selling new ideas, and learning about the needs of complicated software solutions.

For designers and developers to quickly and easily whip up prototypes, the tool's interface must be clear and uncomplicated. The development team requires a tool that is compatible with their preferred operating system as well as others (Windows, macOS, Linux).Dynamic and Interactive Elements The prototyping tool must include dynamic elements like animations and transitions in order to effectively reflect user interactions.

Creative Visual Ability: It should provide a collection of widgets, UI components, and templates for designers to use in making polished prototypes.

Cooperation and information sharing instruments: During the prototyping process, team members should be able to easily communicate and share prototypes with others for feedback.

Ability to Work in Existing Design and Development Environments: The prototyping tool should prioritise integration with other design and development tools such as picture editing software, text editors, and version control systems. The prototype can make use of existing assets, hence the programme must be able to read and write a wide variety of file formats (such as images, vector drawings, and motion pictures).

- Prototype is the creation of a preliminary model or version of a major subsystem or a small, scaled-down version of the entire system
- Prototyping aids in understanding the requirements of complex software products, demonstrating a concept, marketing new ideas, and so on.
- The important features of a prototyping CASE tool are as follows:
- Define user interaction
- Define the system control flow
- Store and retrieve data required by the system

Features of Good Prototyping CASE Tool

- A tool that integrates with the data dictionary can make use of the entries in the data dictionary, help in populating the data dictionary and ensure the consistency between the design data and the prototype.
- A good prototyping tool should support the following features:
- The user should be allowed to define all data entry forms, menus and controls.
- It should properly integrate with the data dictionary of a CASE environment
- Work in manageable modules and modify in successive iterations
- A good prototyping tool should support the following features:
- It should be able to integrate with external user defined modules written in C or some popular high level programming languages.
- The user should be able to define the sequence of states through which a created prototype can run.
- The user should be allowed to control the running of the prototype.
- The run time system of prototype should support mock runs of the actual system and management of the input and output data.
- Work in manageable modules and modify in successive iterations

Type of Prototypes

Each of several ideas of prototyping can be usefully applied in a particular scenario, therefore there is no need to try to synthesise all the uses into one definition or to prescribe one correct approach to the somewhat controversial topic of prototyping. In order to visualise and test

many parts of a product, system, or interface before the final product is built, prototypes are created. Different prototypes are used for different things at different stages of product development.

- Patched-up prototype – all necessary features but not efficient.: The first type of prototyping involves building a functional but hacked-together system. Breadboarding is a technique used in engineering that entails cobbling together a functional model of a (typically) tiny integrated circuit.
- Non operational prototype – only input/output are prototyped.: A nonfunctional, scale model of the final product is the second definition of a prototype. Full-size models of cars are utilised in wind tunnel testing as an illustration of this method. The car's proportions and design are spot-on, but it won't start. Here, only those parts of the car that are necessary for the wind tunnel tests have been included.
- First-of-a-series prototype (beta version)– first full-scale model or pilot:Developing a first working model of a system at full scale is a third definition of prototyping. The first plane in a series could be made into a prototype to test its viability before the construction of the second plane in the series. The prototype is fully functional and represents the designer's vision for a fleet of planes with the same specifications.
- Selected features prototype –operational model with some of the features in a module A fourth definition of prototyping is the creation of a working model that represents the system in part but not in whole. Imagine a brand-new shopping mall that opens before all of the stores have been built.

Selected Features of Prototype: This is usually done to test the independent feature of the software. This is another form of releasing a software in beta version. However, instead of giving the public the full version of the software in beta, only selected features or limited access to some important tools in the program is introduced.

Selected Features Prototype is applied to software that are part of a bigger suite of programs. Those released are independent of the suite but the full version should integrate with other software

The goal of the diagrammatic notation known as Structured Analysis and Structured Design (SA/SD) is to better explain how a system works. The primary purpose of SA/SD is to boost product quality and lower failure probability. It sets out firm guidelines for management and provides supporting paperwork. The system's stability, adaptability, and maintainability are the major points of attention. Some of functions that these diagrams tool support are simple but are very communicative as far as representations of the information of the analysis and design phases are concerned

- Various diagramming tools are used for structured analysis and structured design.
- A CASE tool must support one or more of the structured analysis and design techniques.
- It should aid drawing analysis and design diagrams effortlessly.
- It should support drawing for fairly complex diagrams via a hierarchy of levels.
- The CASE tool should provide easy navigation through the different levels

Testcase Generation CASE Tool:

Automated test case generation is made possible with the help of test case generation tools, which can take the shape of either standalone programmes or larger platforms. Automated test cases can be generated by these instruments using diverse methods like code analysis, model-based testing, and AI-driven algorithms. The following are examples of popular test case generators:

Testing tools can help in automated unit testing, functional regression testing, and performance testing.The CASE tool for test case generation should have the following features:

- It should support design and requirement testing both.
- It should create test set reports in ASCII format so that it can be directly imported into the test plan document.
- A testing tool assures the application readiness, high visibility of test information and types of common defects.
- Integrate with other third party testing tools must be possible.

- It should aid local and remote test execution

CASE Tools and Requirement Engineering:

Requirements definition and management best practices are essential components of any excellent requirements engineering tool. Requirements Engineering requires a highly iterative approach with the end objective of generating well-managed and successful communication and collaboration. Therefore, from a requirements engineering perspective, the following capabilities are essential in a CASE tool: a flexible, feature-rich editor where teams can record and organize needs for the sake of establishing a unified archive task-driven workflow development for issue tracking and change management

The applications, platforms, and programming languages supported by these test case creation tools are extensive. The testing team's preferences, the application's technological stack, and other considerations all play a role in determining which tool to use.

An effective and good requirements engineering tool needs to incorporate the best practices of requirements definition and management

Highly iterative approach with the goal of establishing well managed and effective communication and collaboration must be followed by the Requirements Engineering. Therefore, a CASE tool must have the following features from the viewpoint of requirements engineering, a dynamic, rich editing environment for team members to capture and manage requirements , to create a centralized repository and to create task-driven workflow to do change management, and defect tracking

Four step Process for Requirement Engineering:

1. Requirement Elicitation:
2. Requirement specification
3. Requirement Validation
4. Requirement Management

1. Commonly used elicitation processes are the stakeholder meetings or interviews.

For example, an important first meeting could be between software engineers and customers where they discuss their perspective of the requirements. In requirements engineering, requirements elicitation is the procedure of researching and discovering the requirements of a system from different stake holders like users, customers, etc The practice is also referred to as "requirement gathering". A simple method for requirements elicitation is to ask “WHY”. CASE tools support a dynamic, yet intuitive, requirements capture and management environment that supports content and its editing

2. Requirement Specification: The automated approach of software development needs to have a spirit of collaboration, bringing together world-renowned principal investigators, business analysts, and software development teams.Following expectations from the tool :
   - It should help in developing a properly labelled requirements document that aids in traceability of requirements.
   - It should create both functional and non-functional requirements with related quality attributes. A well-labeled requirements document that aids in requirement traceability is a must. Ideally, you'd make a list of criteria, both functional and non-functional, and the quality attributes that go along with
   - Tracing from user needs to system needs is only one example of the many levels of requirements that may be managed with the help of CASE tools. This guarantees that the completed system meets all specifications.
3. Requirement Validation: To obtain a prioritised and validated approved requirements documentation. Requirements can be checked for completeness, consistency, and alignment with stakeholder needs using the validation features included into CASE systems. Methods like requirement reviews and validation checklists could be used for this purpose. CASE systems typically have validation features that may be used to ensure that the requirements are full, consistent, and meet the demands of all stakeholders. Checklists for validation and requirement reviews are two such approaches.

4. Requirement Management: Requirements can be represented more clearly and intuitively in diagrams and models with the help of visual modelling features offered by some CASE systems. Requirements Reusability can be made With the use of CASE technologies, teams can easily create requirement templates and libraries to share and reuse requirements throughout projects. Management of Documentation for needs using CASE tools facilitate the management and organisation of requirements documentation, making it accessible and searchable by all parties involved.

5. Working Together and Talking to Each Other is called as Collaborating on requirements, giving and receiving feedback, and having open lines of communication are all facilitated by CASE technologies. Collaborative editing, commenting, and notification systems are all examples of such capabilities.

6. Requirements can be stored and managed in one convenient location with the help of most CASE solutions. This guarantees that the most up-to-date set of requirements is available to everyone involved. In order to ensure that the final software product lives up to the requirements and expectations of all parties involved, development teams can benefit from using CASE techniques in requirement engineering

SELF-ASSESSMENT QUESTIONS – 4

9. _____ control the behavior of tools within the environment.
10. _____ is a database that acts as the center for both accumulation and storage of software engineering information.

7. SUMMARY

Let's recapitulate the important points discussed in this unit.

- Computer-aided software engineering tools span every activity in the software process and those umbrella activities that are applied throughout the process.
- CASE combines a set of building blocks that begins at the hardware and operating system software level and end with individual tools.
- CASE categories encompass both management and technical activities that span most software application areas. Each category of tool is considered a "point solution."
- The I-CASE environment combines integration mechanisms for data, tools, and human/computer interaction.
- Data integration can be achieved through direct exchange of information, through common file structures, by data sharing or interoperability, or through the use of a full I-CASE repository.
- Tools integration can be custom designed by vendors who work together or achieved through management software provided as part of the repository. Human/computer integration is achieved through interface standards that have become commonplace throughout the industry. Integration architecture is designed to facilitate the integration of users with tools, tools with tools, tools with data, and data with data.
- The CASE repository has been referred to as a "software bus." Information moves through it, passing from tool to tool as software engineering progresses. But the repository is much more than a "bus." It is also a storage place that combines sophisticated mechanisms for integrating CASE tools and thereby improving the process through which software is developed. The repository is a relational or object-oriented database that is "the center of accumulation and storage" for software engineering.

8. TERMINAL QUESTIONS

1. Explain the classifications of CASE tools.
2. List out the necessary steps for implementing CASE tools.
3. What do you mean by I-CASE? Explain.
4. Give the architecture of CASE environment.

9. ANSWERS

Self-Assessment Questions

1. True
2. Computer Aided Software Engineering
3. a) Repository
4. True
5. Process modeling tools
6. b) Prototyping
7. True
8. Software Configuration Items
9. Tools management services
10. CASE Repository

Terminal Questions

1. Business Process Engineering Tools, Process Modeling and Management Tools, Project Planning Tools, Risk Analysis Tools, Project Management Tools, etc. (Refer Section 3 for detail)
2. Conduct a technology-impact study to determine how the basic business of the organization should change to maximize the opportunities presented by rapid technological change. Evaluate how the organization should be re-engineered to take advantage of new technology. (Refer Section 4)
3. An I-CASE environment is a collection of CASE tools and other components together with an integration approach that supports most or all of the interactions that occur among the environment components, and between the users of the environment and the environment itself. (Refer Section 5)
4. A software engineering team uses CASE tools, corresponding methods, and a process framework to create a pool of software engineering information. The integration framework facilitates transfer of information into and out of the pool. (Refer Section 6)