SOFTWARE ENGINEERING UNIT-IV PART-II

Computer Aided Software Engineering: Case and its Scope, Case Environment, Case Supporting Software Life Cycle, Other Characteristics of Case Tools, Towards Second Generation CASE Tool, Architecture of a Case Environment

CASE AND ITS SCOPE

CASE (Computer Aided Software Engineering) is a tool support for software development, management and maintenance activities. The use of CASE tools cases and reduces the development, management, and maintenance effort. Automated tools accelerate the development activities. The CASE tools help the developers, managers and other key personnel their productivity in the development team. Some of these CASE tools assist in phase related tasks such as specification, structured analysis, design, coding, testing, etc.; and others to non-phase activities such as project management and configuration management. The primary reasons for using a CASE tool are:

- To increase productivity
- To help produce better quality software at lower cost

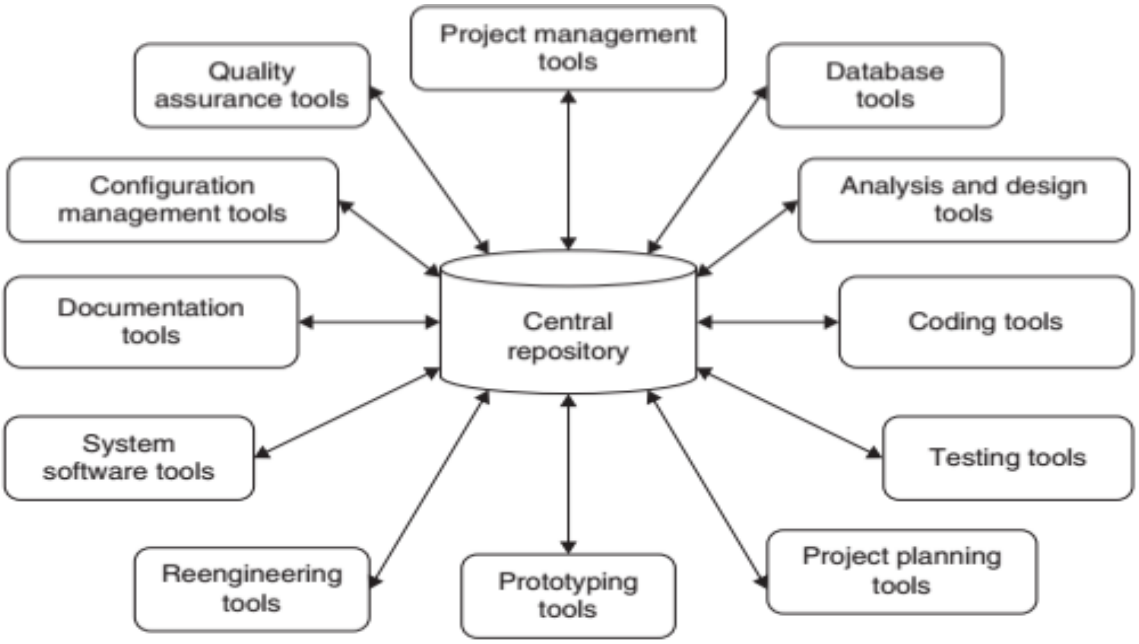
CASE ENVIRONMENT

Software development life cycle activities can be assisted by the use of software tools. Programmers can use automated tools for development, maintenance, management, and planning purposes. The CASE environment of various automated tools is shown in figure. Some of the widely used tools, documentation tools, system software tools, quality assurance tools, database tools, software configuration management tools, prototyping tools, coding tools, testing tools, reengineering tools etc.

Project management tools-Management tools project management tools can be used to design project schedule, activity plan, etc.

Database management tools- The database for relational and object-oriented database management system is designed to provide support for the CASE repository.

Analysis and design the tools –These tools help to produce analysis and design models of the system to be built. These models describe the problem and the solutions of the system in the context.



Programming tools- Compilers, editors debuggers, programming environments, fourth generation languages, graphical programming environments, applications generators, and database query generators are the programming tools used for programming purposes.

Integration and testing tools- These tools are basically used for data preparation, static and dynamic measurements, simulation, and test planning.

Project planning tools- These tools are useful for cost and effort estimation and project scheduling activities.

Prototyping tools- These tools help to design screen layouts, data design, and report generation.

Reengineering tools- Reengineering tools are used during system migration from an old platform to a new platform. There are various reengineering tools such as reverse engineering to specification tools, code restricting and analysis tools, and online system reengineering tools.

System software tools- There are various tools that assist to work in desktop and network systems. For example, network system software, object management services, distributed component support, and communications software.

Documentation tools- These tools are useful to produce documentation of the work products that pave the software development process.

Software configuration management tools- The configuration management tools assist configuration management activities such as configuration identification, version control, change control, auditing, status accounting, etc.

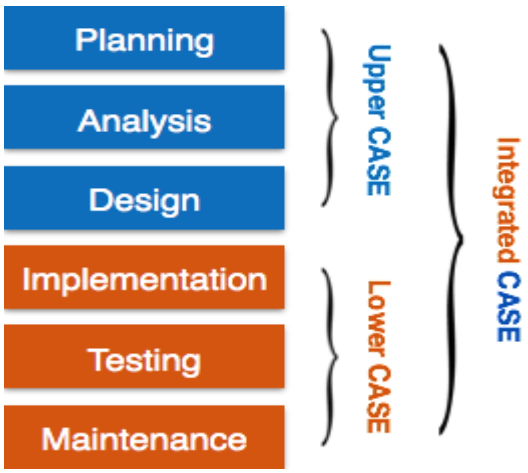
Quality assurance tools- The quality assurance tools are the auditing tools used to determine compliance with language standards or to ensure the quality of software being built.

CASE SUPPORTING SOFTWARE LIFE CYCLE

CASE supports different software engineering activities within a SDLC process. CASE tools are designed to improve the quality and upgrade the computer system. It provides computerized settings to the developers to analyze problems and then design its system model. The SDLC process is very expensive and sometime the project become more complex in nature so its more difficult for the implementation, in this situation the developers looking for such CASE tools that helps in many different development phases of the software.

CASE tools can be broadly divided into the following parts based on their use at a particular SDLC stage:

- **Central Repository** - CASE tools require a central repository, which can serve as a source of common, integrated and consistent information. Central repository is a central place of storage where product specifications, requirement documents, related reports and diagrams, other useful information regarding management is stored. Central repository also serves as data dictionary.



- **Upper Case Tools** - Upper CASE tools are used in planning, analysis and design stages of SDLC.

- **Lower Case Tools** - Lower CASE tools are used in implementation, testing and maintenance.
- **Integrated Case Tools** - Integrated CASE tools are helpful in all the stages of SDLC, from Requirement gathering to Testing and documentation.

CASE tools can be grouped together if they have similar functionality, process activities and capability of getting integrated with other tools.

CHARACTERISTICS OF CASE TOOLS

A CASE tool must have the following characteristics in order to be used efficiently:

- **A standard methodology:** A CASE tool must support a standard software development methodology and standard modeling techniques. In the present scenario most of the CASE tools are moving towards UML.
- **Support at Project Management:** The CASE tool ought to support assembling, storing, and analyzing data on the computer code project's progress like the calculable task length, regular and actual task begin, completion date, dates, and results of the reviews, etc.
- **Flexibility:** Flexibility in use of editors and other tools. The CASE tool must offer flexibility and the choice for the user of editors' development environments.
- **Strong Integration:** The CASE tools should be integrated to support all the stages. This implies that if a change is made at any stage, for example, in the model, it should get reflected in the code documentation and all related design and other documents, thus providing a cohesive environment for software development.
- **Integration with testing software:** The CASE tools must provide interfaces for automatic testing tools that take care of regression and other kinds of testing software under the changing requirements.
- **Support for reverse engineering:** A CASE tools must be able to generate complex models from already generated code.
- **On-line help:** The CASE tools provide an online tutorial.

TOWARDS SECOND GENERATION CASE TOOL

As far as code generation is concerned, the general expectation of a CASE tool is quite low. A reasonable requirement is traceability from source file to design data. More pragmatic supports expected from a CASE tool during code generation phase are the following:

- The CASE tool should support generation of module skeletons or templates in one or more popular languages. It should be possible to include copyright message, brief description of the module, author name and the date of creation in some selectable format.
- The tool should generate records, structures, class definition automatically from the contents of the data dictionary in one or more popular languages.
- It should generate database tables for relational database management systems.
- The tool should generate code for user interface from prototype definition for X window and MS window based applications.

Test Case Generation Case Tool

The CASE tool for test case generation should have the following features:

- It should support both design and requirement testing.
- It should generate test set reports in ASCII format which can be directly imported into the test plan document.

An important feature of the **second-generation CASE tool** is the direct support of any adapted methodology. This would necessitate the function of a CASE administrator organization who can tailor the CASE tool to a particular methodology. In addition, the second-generation CASE tools have following features:

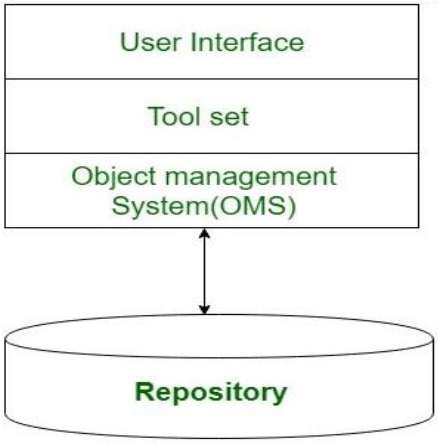
- Intelligent diagramming support- The fact that diagramming techniques are useful for system analysis and design is well established. The future CASE tools would provide help to aesthetically and automatically lay out the diagrams.
- Integration with implementation environment- The CASE tools should provide integration between design and implementation.
- Data dictionary standards- The user should be allowed to integrate many development tools into one environment. It is highly unlikely that any one vendor will be able to deliver a total solution. Moreover, a preferred tool would require tuning up for a

particular system. Thus the user would act as a system integrator. This is possibly only if some standard on data dictionary emerges.

- Customization support- The user should be allowed to define new types of objects and connections. This facility may be used to build some special methodologies. Ideally it should be possible to specify the rules of a methodology to a rule engine for carrying out the necessary consistency checks.

ARCHITECTURE OF A CASE ENVIRONMENT

The architecture of a typical modern CASE environment is shown diagrammatically in below figure. The important components of a modern CASE environment are user interface, tool set, object management system (OMS), and a repository. Characteristics of a tool set have been discussed earlier.



❖ User Interface

The user interface provides a consistent framework for accessing the different tools thus making it easier for the users to interact with the different tools and reducing the overhead of learning how the different tools are used.

❖ Tool set

The tool set is integrated through the data dictionary which supports multiple projects, multiple users working simultaneously and allows sharing information between users and projects.

❖ Object Management System (OMS) and Repository

Different case tools represent the software product as a set of entities such as specification, design, text data, project plan, etc. The object management system maps these logical entities such into the underlying storage management system (repository). The commercial relational database management systems are geared towards supporting large volumes of information structured as simple relatively short records. There are a few types of entities but large number of instances. By contrast, CASE tools create a large number of entity and relation types with perhaps a few instances of each. Thus the object management system takes care of appropriately mapping into the underlying storage management system.