# Module 15

# Computer Aided Software Engineering

# Lesson 38

# Different Characteristics of CASE Tools

## Specific Instructional Objectives

At the end of this lesson the student would be able to:

- Identify hardware and environmental requirements for a CASE tool.
- Identify the software documentation support that might be available from CASE tools.
- Identify the two project management supports that should be available from CASE tools.
- Identify the support that needs to be provided by CASE tools for interfacing with other CASE tools.
- Identify the two software reverse engineering supports that should be available from CASE tools.
- Identify the wo features that might be supported by data dictionary interface of a CASE environment.
- Identify the non-traditional features supported by second generation CASE tools.
- Explain the architecture of a CASE environment with the help of a suitable diagram.

## Hardware and environmental requirements

In most cases, it is the existing hardware that would place constraints upon the CASE tool selection. Thus, instead of defining hardware requirements for a CASE tool, the task at hand becomes to fit in an optimal configuration of CASE tool in the existing hardware capabilities. Therefore, it can be emphasized on selecting the most optimal CASE tool configuration for a given hardware configuration.

The heterogeneous network is one instance of distributed environment and this can be chosen for illustration as it is more popular due to its machine independent features. The CASE tool implementation in heterogeneous network makes use of client-server paradigm. The multiple clients who run different modules access data dictionary through this server. The data dictionary server may support one or more projects. Though it is possible to run many servers for different projects but distributed implementation of data dictionary is not common.

The tool set is integrated through the data dictionary which supports multiple projects, multiple users working simultaneously and allows to share information between users and projects. The data dictionary provides consistent view of all project entities, e.g. a data record definition and its entity-relationship diagram be consistent. The server should depict the per-project logical view of the data dictionary. This means that it should allow back up/restore, copy, cleaning part of the data dictionary, etc.

The tool should work satisfactorily for maximum possible number of users working simultaneously. The tool should support multi-windowing environment for the users. This is important to enable the users to see more than one diagram at a time. It also facilitates navigation and switching from one part to the other.

## Documentation support

The deliverable documents should be organized graphically and should be able to incorporate text and diagrams from the central repository. This helps in producing up-to-date documentation. The CASE tool should integrate with one or more of the commercially available desktop publishing packages. It should be possible to export text, graphics, tables, data dictionary reports to the DTP package in standard forms such as PostScript.

## Project management support

The CASE tool should support collecting, storing, and analyzing information on the software project's progress such as the estimated task duration, scheduled and actual task start, completion date, dates and results of the reviews, etc.

## External interface

The CASE tool should allow exchange of information for reusability of design. The information which is to be exported by the CASE tool should be preferably in ASCII format and support open architecture. Similarly, the data dictionary should provide a programming interface to access information. It is required for integration of custom utilities, building new techniques, or populating the data dictionary.

## Reverse engineering

The CASE tool should support generation of structure charts and data dictionaries from the existing source codes. It should populate the data dictionary from the source code. If the tool is used for re-engineering information systems, it should contain conversion tool from indexed sequential file structure, hierarchical and network database to relational database systems.

## Data dictionary interface

The data dictionary interface should provide view and update access to the entities and relations stored in it. It should have print facility to obtain hard copy of the viewed screens. It should provide analysis reports like cross-referencing, impact analysis, etc. Ideally, it should support a query language to view its contents.

## Second-generation CASE tool

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 fig. 15.2. 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.

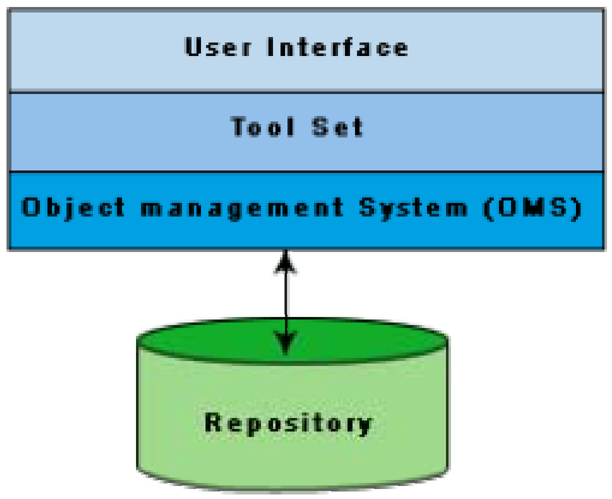

**Fig. 15.2:** Architecture of a Modern CASE Environment

**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.

**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.

The following questions have been designed to test the objectives identified for this module:

1. What do you understand by the term CASE tool?

2. What are the primary objectives of a CASE tool?

3. What do you understand by the term CASE environment?

4. Differentiate between the characteristics of a CASE environment and a programming environment.

5. What are the main advantages of using CASE tools?

6. Discuss the role of the data dictionary in a CASE environment.

7. What features are supported by a good prototyping CASE tool?

8. Discuss the supports available form CASE tools in order to perform structured analysis and software design activity.

9. During code generation what supports might be expected from CASE tools?

10. How can CASE tool help for the purpose of test case generation?

11. Discuss hardware and environmental requirements for a CASE tool.

12. What are the software project management supports that might be expected from CASE tools?

13. What are the software reverse engineering supports that might be available from CASE tools?

14. What are some of the important features that a future generation CASE tool should support?

15. Schematically draw the architecture of a CASE environment and explain how the different tools are integrated.

## Mark all options which are true.

1. Computer Aided Software Engineering (CASE) tools can assist in

   □ phase related activities such as specification, structured analysis, coding, testing, etc.
   □ non-phase related activities such as software project management, software configuration management, etc.
   □ neither phase related activities nor non-phase related activities

2. The primary objective(s) in using any CASE tool is(are):

   □ to increase productivity of software development
   □ to decrease software development as well as software maintenance cost

□ to help produce better quality software
□ all of the above

3. Which of the following features are related to a prototyping CASE tool?

□ to define user interaction
□ to define the control flow of the system
□ to incorporate some processing logic
□ all of the above

4. Which of the following supports should we expect from a CASE tool during the code generation phase of a software development project?

□ generation of module skeletons or templates in one or more popular languages
□ generation of records, structures, class definition automatically from the contents of the data dictionary in one or more popular languages
□ generation of database tables for relational database management systems
□ all of the above

5. Which of the following features that are not present in current CASE tools but are likely to be supported as a second-generation CASE tool?

□ diagramming support
□ documentation support
□ customization support for different development methodologies
□ querying data dictionary

## Mark the following statements as either True or False. Justify your answer.

1. A programming environment is an integrated collection of tools supporting the activities related to all the phases of software development life cycle.

2. Use of CASE tools typically leads to improvements to the quality of a software product.

3. CASE tools help in producing c