**CASE** 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.

**CASE Tool**

1. A CASE tool is a computer-based product aimed at supporting one or more software engineering activities within a software development process.

2. Computer-Aided Software Engineering tools are those software which are used in any and all phases of developing an information system, including analysis, design and programming. For example, data dictionaries and diagramming tools aid in the analysis and design phases, while application generators speed up the programming phase.

3. CASE tools provide automated methods for designing and documenting traditional structured programming techniques. The ultimate goal of CASE is to provide a language for describing the overall system that is sufficient to generate all the necessary programs needed.

**CLASSIFICATION of CASE TOOLS**

Existing CASE tools can be classified along 4 different dimensions:

1. Life-cycle support

2. Integration dimension

3. Construction dimension

4. Knowledge-based CASE dimension

Let us take the meaning of these dimensions along with their examples one by one:

**Life-Cycle Based CASE Tools**

This dimension classifies CASE Tools on the basis of the activities they support in the information systems life cycle. They can be classified as Upper or Lower CASE tools.

**UpperCASE Tool** UpperCASE Tool is a Computer-Aided Software Engineering (CASE) software tool that supports the software development activities upstream from implementation. Uppercasetool focus on the analysis phase (but sometimes also the design phase) of the software development lifecycle (diagramming tools, report and form generators, and analysis tools)

 **LowerCASE Tool** LowerCASE Tool Computer-Aided Software Engineering (CASE) software tool that directly supports the implementation (programming) and integration tasks. LowerCASE tools support database schema generation, program generation, implementation, testing, and configuration management.

**Integration dimension**

Three main CASE Integration dimensions have been proposed:

1. CASE Framework

2. ICASE Tools Tools that integrate both upper and lower CASE, for example making it possible to design a form and build the database to support it at the same time. An automated system development environment that provides numerous tools to create diagrams, forms and reports. It also offers analysis, reporting, and code generation facilities and seamlessly shares and integrates data across and between tools.

3. Integrated Project Support Environment(IPSE)

**Types of CASE Tools** The general types of CASE tools are listed below:

1. **Diagramming tools:** enable system process, data and control structures to be represented graphically.

2. **Computer display and report generators:** help prototype how systems look and feel. It makes it easier for the systems analyst to identify data requirements and relationship.

3. **Analysis tools:** automatically check for importance, inconsistent, or incorrect specifications in diagrams, forms, and reports.

4. **Central repository:** enables the integrated storage of specifications, diagrams, reports and project management information.

5. **Documentation Generators:** produce technical and user documentation in standard formats.

6. **Code generators:** enable the automatic generation of program and data base definition code directly from the design documents, diagrams, forms, and reports.

**Functions of a CASE Tool**

1. **Analysis** CASE analysis tools automatically check for incomplete, inconsistent, or in correct specifications in diagrams, forms and reports.

2. **Design** This is where the technical blueprint of the system is created by designing the technical architecture – choosing amongst the architectural designs of telecommunications, hardware and software that will best suit the organization’s system and future needs. Also designing the systems model – graphically creating a model from graphical user interface, screen design, and databases, to placement of objects on screen

3. **Code generation** CASE Tool has code generators which enable the automatic generation of program and data base definition code directly from the documents, diagrams, forms, and reports.

4. **Documentation** CASE Tool has documentation generators to produce technical and user documentation in standard forms. Each phase of the SDLC produces documentation. The types of documentation that flow from one face to the next vary depending upon the organization, methodologies employed and type of system being built.

**CASE Environments**

An environment is a collection of CASE tools and workbenches that supports the software process. CASE environments are classified based on the focus/basis of integration

1. Toolkits

2. Language-centered

3. Integrated

4. Fourth generation

5. Process-centered

**Toolkits**

Toolkits are loosely integrated collections of products easily extended by aggregating different tools and workbenches. Typically, the support provided by a toolkit is limited to programming, configuration management and project management. And the toolkit itself is environments extended from basic sets of operating system tools, for example, the Unix Programmer's Work Bench and the VMS VAX Set. In addition, toolkits' loose integration requires user to activate tools by explicit invocation or simple control mechanisms. The resulting files are unstructured and could be in different format, therefore the access of file from different tools may require explicit file format conversion. However, since the only constraint for adding a new component is the formats of the files, toolkits can be easily and incrementally extended.

**Language-centered**

The environment itself is written in the programming language for which it was developed, thus enabling users to reuse, customize and extend the environment. Integration of code in different languages is a major issue for language-centered environments. Lack of process and data integration is also a problem. The strengths of these environments include good level of presentation and control integration. Interlisp, Smalltalk, Rational, and KEE are examples of language-centered environments.

**Integrated**

These environments achieve presentation integration by providing uniform, consistent, and coherent tool and workbench interfaces. Data integration is achieved through the*repository* concept: they have a specialized database managing all information produced and accessed in the environment. Examples of integrated environment are the ICL CADESsystem, IBM AD/Cycle and DEC Cohesion.

**Fourth-generation**

Fourth-generation environments were the first integrated environments. They are sets of tools and workbenches supporting the development of a specific class of program: electronic data processing and business-oriented applications. In general, they include programming tools, simple configuration management tools, document handling facilities and, sometimes, a code generator to produce code in lower level languages. Informix 4GL, and Focus fall into this category.

**Process-centered**

Environments in this category focus on process integration with other integration dimensions as starting points. A process-centered environment operates by interpreting a process model created by specialized tools. They usually consist of tools handling two functions:

 Process-model execution

 Process-model production

Examples are East, Enterprise II, Process Wise, Process Weaver, and Arcadia

|  |  |
| --- | --- |
| **CASE Tools ADVANTAGES** | **DISADVANTAGES** |
| Helps standardization of notations and diagrams | Limitations in the flexibility of documentation |
| Help communication between development team members | May lead to restriction to the tool's capabilities |
| Automatically check the quality of the models | Major danger: completeness and syntactic correctness does NOT mean compliance with requirements |
| Reduction of time and effort | Costs associated with the use of the tool: purchase + training |
| Enhance reuse of models or models' components | Staff resistance to CASE tools |