**What are CASE Tools?**

**Computer-aided software engineering (CASE) tools assist software engineering managers and practitioners in every activity associated with the software process. They automate project management activities, manage all work products produced throughout the process, and assist engineers in their analysis, design, coding and testing work. CASE tools can be integrated within a sophisticated environment.**

**CASE is the use of computer-based support in the software development process; a CASE tool is a computer-based product aimed at supporting one or more software engineering activities within a software development process; a 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.**

**The goal of introducing CASE tools is the reduction of the time and cost of software development and the enhancement of the quality of the systems developed. The interest in CASE tools and environments is based on expectations about increasing productivity, improving product quality, facilitating maintenance, and making software engineers' task less odious and more enjoyable.**

**CASE Tools and its scope**

**CASE technology is the automation of step by step methodologies for software and system development. CASE tools are characterized by the stage or stages of software development life cycle on which they focus. Since different tools covering different stages share common information, it is required that they integrate through some central repository system (data dictionary) to have a consistent view of such information. In phases of software development life cycle integrated through a central data dictionary. Case Tools are used in many ways in our organizations. Case tools can be broadly classed into these broader areas:**

* **Requirement Analysis Tool**
* **Structure Analysis Tool**
* **Software Design Tool**
* **Code Generation Tool**
* **Test Case Generation Tool**
* **Document Production Tool**
* **Reverse Engineering Tool**

**While many organizations still use the SDLC methodology, it is often supplemented with other methods. Many systems developers use the CASE tools in various stages of the Software Development Life Cycle. They mainly use it while developing the following methodologies:**

* **Life Cycle**
* **Object-oriented Approach**
* **Rapid Applications Development (RAD)**
* **Prototyping**
* **Joint Applications Development (JAD)**

**The job of a systems developer may contain requirements analysis, process design, data design, and programming among other activities. But, not all systems developers do the same activities. One may spend most of his or her time on analysis; another, on design. The various activities that the system developers involve include Systems Analysis (including feasibility studies and requirements definition), Systems Design (including user interface, data, and process design), Programming (or generating code),**

**Testing, Supervisory or other management tasks and Maintenance. CASE tools play an important role in helping the system developers to perform the task efficiently.**

**Classification**

**An upper CASE tool (front end CASE) provides support for the early stages in the systems development life cycle such as requirements analysis and design. A lower CASE tool (back end CASE) provides support for the later stages in the life cycle such as code generation and testing. Integrated CASE tools support both the early and later stages. Further classifications usually list which functionalities are supported by the tool, such as data flow diagrams, entity relationships data models, etc. provides a different type of model of CASE functionality which helps organize CASE tools.**

**Upper CASE Tools support business and analysis modeling. They support traditional diagrammatic languages such as ER diagrams, Data flow diagram, Structure charts, Decision Trees, Decision tables, etc. Lower CASE Tools support development activities, such as physical design, debugging, construction, testing, component integration, maintenance, and reverse engineering. All other activities span the entire life-cycle and apply equally to upper and lower CASE.**

**A. Fuggetta classified CASE software into 3 categories:**

1. ***Tools* support specific tasks in the software life-cycle.**
2. ***Workbenches* combine two or more tools focused on a specific part of the software life-cycle.**
3. ***Environments* combine two or more tools or workbenches and support the complete software life-cycle.**

**CASE tools supports specific tasks in the software development life-cycle. They can be divided into the following categories:**

1. **Business and Analysis modeling. Graphical modeling tools. E.g., E/R modeling, object modeling, etc.**
2. **Development. Design and construction phases of the life-cycle. Debugging environments. E.g.,**[**GNU Debugger**](https://en.wikipedia.org/wiki/GNU_Debugger)**.**
3. **Verification and validation. Analyze code and specifications for correctness, performance, etc.**
4. **Configuration management. Control the check-in and check-out of repository objects and files. E.g.,**[**SCCS**](https://en.wikipedia.org/wiki/Source_Code_Control_System)**, CMS.**
5. **Metrics and measurement. Analyze code for complexity, modularity (e.g., no "go to's"), performance, etc.**
6. **Project management. Manage project plans, task assignments, scheduling.**

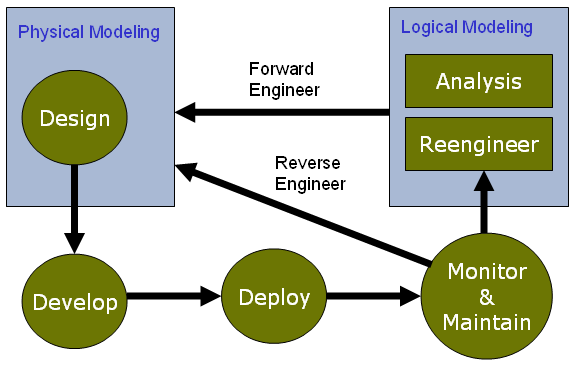
**An example workbench is Microsoft's Visual Basic programming environment. It incorporates several development tools: a GUI builder, smart code editor, debugger, etc. Most commercial CASE products tended to be such workbenches that seamlessly integrated two or more tools. Workbenches also can be classified in the same manner as tools; as focusing on Analysis, Development, Verification, etc. as well as being focused on upper case, lower case or processes such as configuration management that span the complete life-cycle.**

**List of CASE Tools**

[**http://www.rspa.com/spi/case.html#compendia**](http://www.rspa.com/spi/case.html#compendia)

**Forward Engineering** Going from a logical data model to a physical data model. This is easy because the design includes all dependencies, indexes and relationships between the components of the data model.

**Reverse Engineering** Attempting to reconstruct the logical data model from a physical data model. This is hard because not every database engine has the means to store the interdependencies between objects in a logical model, and sometimes these relationships are lost altogether. This information has to be somehow recovered by analyzing the data and inferring the missing relationships.



### **Team Leaders:**

**Project management is a people-intensive activity, and for this reason, competent practitioners often make poor team leaders. A good team leader should have some abilities described in the MOI model of leadership in below,**

**Jerry Weinberg suggests a MOI model of leadership:**

**Motivation: The ability to encourage technical people to produce to their best ability.**

**Organization: The ability to modify existing processes (or invent new ones) that will enable the initial concept to be translated into a final product.**

**Ideas or innovation: The ability to encourage people to create and feel creative even when they must work within bounds established for a particular software product or application.**