SOFTWARE ENGINEERING
TOOLS AND PRACTICE

Course objectives

- This course provides an introduction to the Software Engineering Tools and Practices –
- a look at a typical approach software engineers create applications in practice. Topics include requirements analysis, high-level design, detail-level design, UML modeling, code generation, application building, and revision management.
- In order to achieve the above objectives
  - create UML class, state, and sequence diagrams using a CASE tool
  - generate source code from UML design models, and synchronize subsequent changes
  - create a deployable software package using an automated build tool
  - create an installable software package using an automated build tool
-

Motivation

- Software engineering tools and practices have a positive influence on the quality and cost of developed software and on software development processes.
- There are many benefits from making these tools and practices integral to software development and acquisition processes, including:
  - Reduced risk: programs deliver more predictably
  - Improved customer satisfaction: products developed experience fewer field defects
  - Lower cost of ownership: programs experience lower life-cycle maintenance costs when deployed operationally

Prerequisites

- Proficiency in a high-level object-oriented programming language
- Knowledge of basic object-oriented programming concepts, data structures, and software design techniques.

Chapter one:
Introduction to software engineering tools and practices

- Contents
  - Introduction
  - Basic of CASE tools

Introduction

- Software can be defined as a set of programs and associated documentations.
  - which includes
    - A number of separate programs
    - Configuration files
    - System documentation
    - User documentation

What is software engineering ?

- Software Engineering can be defined as the construction of *quality* software with *a limited budget* and *a given deadline* in the *context of constant change.*

- Software engineering (SE) is an intellectual activity and thus human-intensive
- Software is built to meet a certain functional goal and satisfy certain qualities
- Software processes also must meet certain qualities

Brainstorm about the development of software engineering

- Try to realize the improvements on computer hardware
- Also about the software
- **But what about the improvements on software engineering**

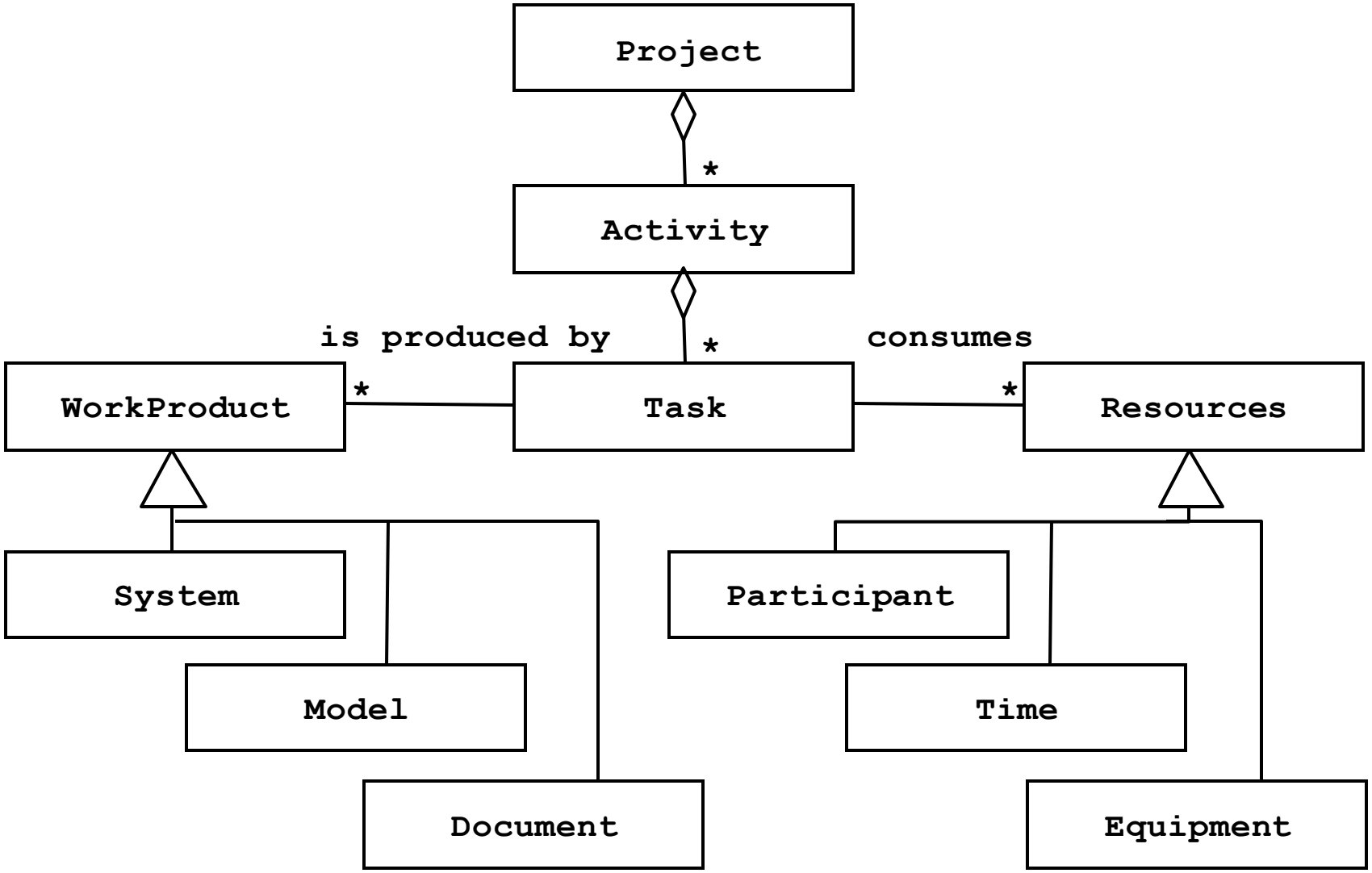
Fundamental assumptions of software engineering

- ✓Good processes lead to good software
- ✓ Good processes reduce risk

Software Engineering Concepts: Definitions

- **Project** – set of activities to develop a software system
- **Activity** – a phase in which related tasks are carried out
- **Task** – effort that uses resources AND produces Work Product
- **Resources** – time, equipment, people (participants)
- **Work Product** – a model, system, or artifact

Software Engineering Concepts



Software Engineering Activities

- ✓Requirements Elicitation
- ✓Analysis
- ✓Object Design
- ✓System Design
- ✓Implementation
- ✓Validation/testing

Systems, Models, and Views

- A *model* is an abstraction describing system or a subset of a system
- A *view* depicts selected aspects of a model
- A *notation* is a set of graphical or textual rules for representing views

- *How views and models of a single system may overlap each other?*

Why model software?

Software is already an abstraction: why model software?

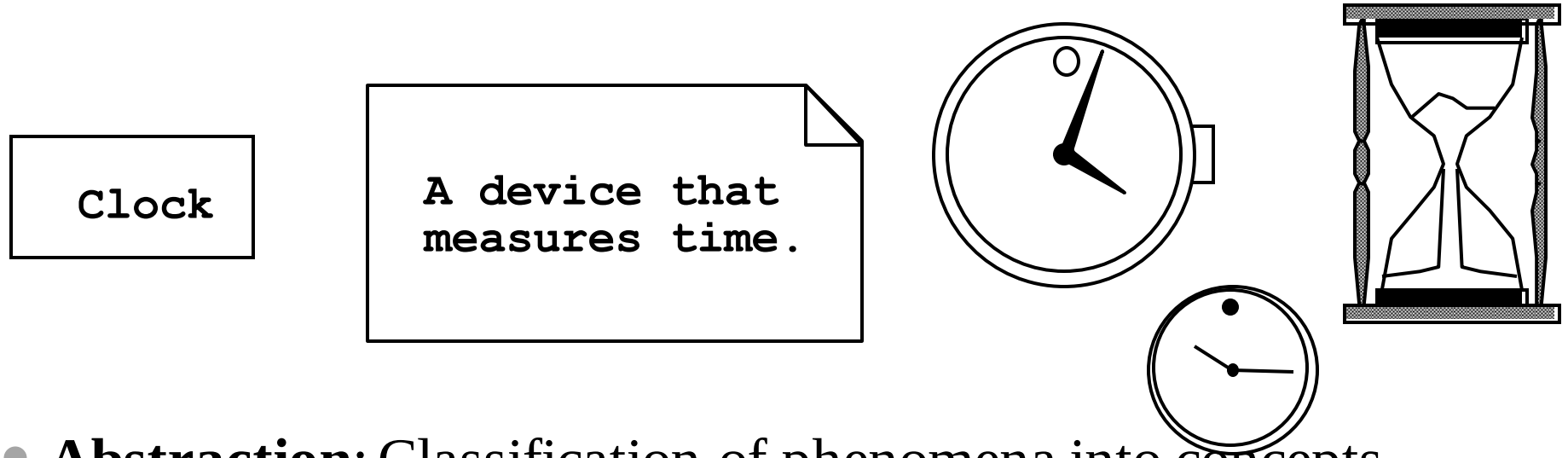
- Software is getting larger, not smaller
  - 5.0 ~ 40 million lines of code
  - A single programmer cannot manage this amount of code in its entirety.
- Code is often not directly understandable by developers who did not participate in the development
- We need simpler representations for complex systems
  - Modeling is a mean for dealing with complexity

Concepts and Phenomena

- ***Phenomenon***: An object in the world of a domain as you perceive it, for example:
  - The lecture you are attending
  - My black watch
- ***Concept***: Describes the common properties of a phenomena, for example:
  - Lectures on software engineering
  - Black watches
- ***A* concept is a 3-tuple**:
  - Its *Name* distinguishes it from other concepts.
  - Its *Purpose* are the properties that determine if a phenomenon is a member of a concept.
  - Its *Members* are the phenomena which are part of the concept.

Concepts and Phenomena

Name **Purpose** **Members**



- **Abstraction**: Classification of phenomena into concepts
- **Modeling**: Development of abstractions to answer specific questions about a set of phenomena while ignoring irrelevant details.

Concepts In Software: Type and Instance

- Type:
  - An abstraction in the context of programming languages
  - Name: int, Purpose: integral number, Members: 0, -1, 1, 2, -2, . . .
- Instance:
  - Member of a specific type
- The type of a variable represents all possible instances the variable can take.
- The relationship between “type” and “instance” is similar to that of “concept” and “phenomenon.”
- Abstract data type:
  - Special type whose implementation is hidden from the rest of the system.

CASE Tools

- CASE stands for **C**omputer **A**ided **S**oftware **E**ngineering. It means, development and maintenance of software projects with help of various automated software tools.

- **Computer-aided software engineering** (**CASE**) is the scientific application of a set of tools and methods to a software system which is meant to result in high-quality, defect-free, and maintainable software products.
- It also refers to methods for the development of information systems together with automated tools that can be used in the software development process

- CASE tools are set of software application programs, which are used to automate SDLC activities. CASE tools are used by software project managers, analysts and engineers to develop software system.
- There are number of CASE tools available to simplify various stages of Software Development Life Cycle such as Analysis tools, Design tools, Project management tools, Database Management tools, Documentation tools are to name a few.
- Use of CASE tools accelerates the development of project to produce desired result and helps to uncover flaws before moving ahead with next stage in software development.

- A kind of component-based development which allows its users to rapidly develop information systems. The main goal of case technology is the automation of the entire information systems development life cycle process using a set of integrated software tools, such as modeling, methodology and automatic code generation.

- Computer-aided software engineering" (CASE) can refer to the software used for the automated development of systems software, i.e., computer code.
- CASE software supports the software process activities such as requirement engineering, design, program development and testing.
- Therefore, CASE tools include design editors, data dictionaries, compilers, debuggers, system building tools, etc.
- CASE also refers to the methods dedicated to an engineering discipline for the development of information system using automated tools.
- CASE is mainly used for the development of quality software which will perform effectively.

- **Characteristics of CASE:**
  - It is a graphic oriented tool.
  - It supports decomposition of process.
- **Some typical CASE tools are:**
  - Unified Modeling Language
  - Data modeling tools, and
  - Source code generation tools

- Why CASE tools?
  - Architecture Management
    - Model, design, and rapidly build Software, Systems, and Computer Application Programs.
  - Change and Release Management
    - Improve software delivery and lifecycle traceability, from requirements through deployment.
  - Software Development Management
    - Align projects for improved productivity and predictability.
  - Quality Management
    - Ensure software functionality, reliability and performance throughout development and production.

- Goal of using CASE tools?
  - Supply basic functionality, do routine tasks automatically
    - Be able to support editing of code in the particular programming language, supply refactoring tools
  - Enhance productivity
    - Generate code pieces automatically
  - Increase software quality
  - Intuitive use
  - Integration with other tools
    - For example, code editor works with code repository

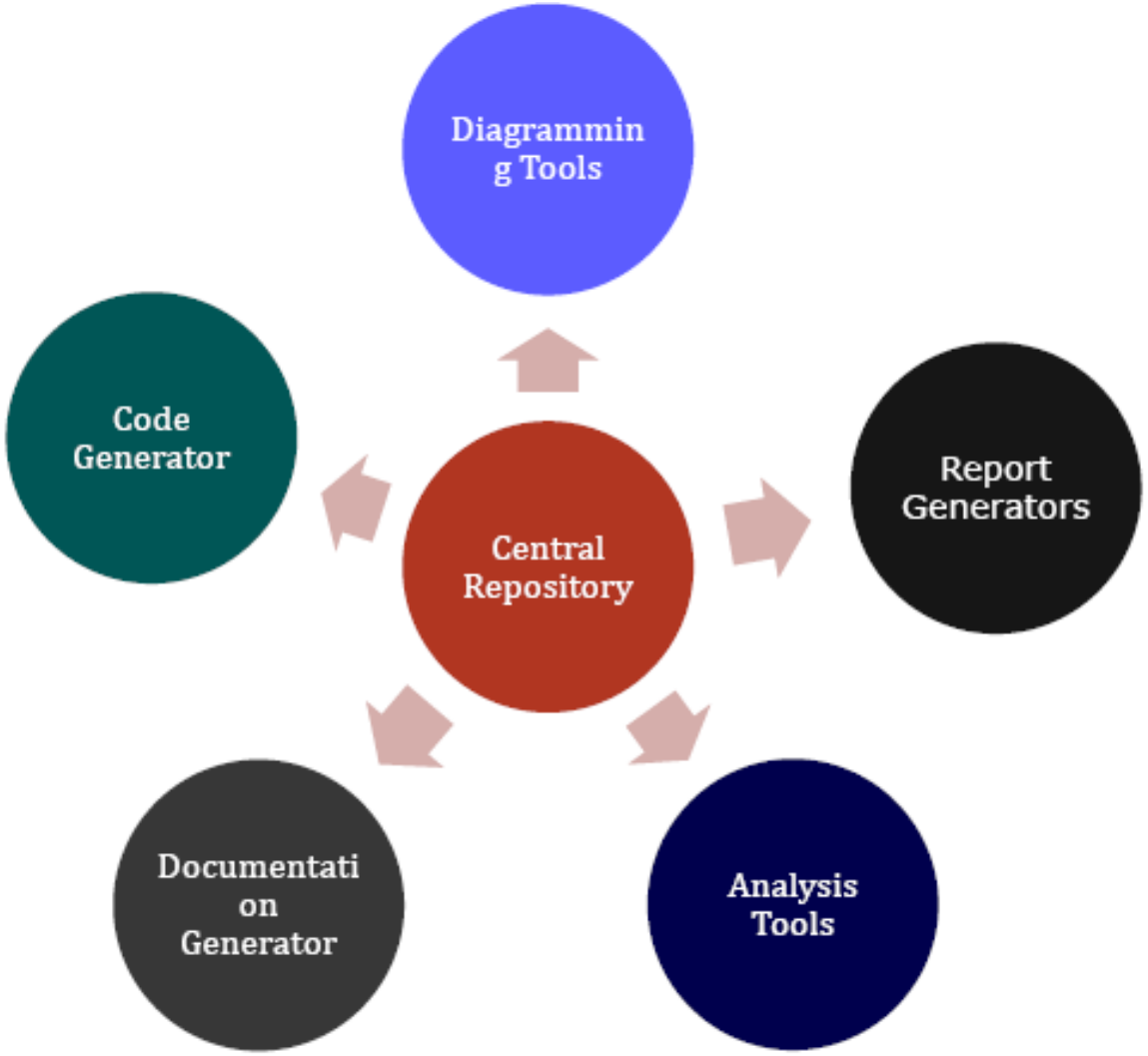
Importance of CASE

- CASE allows for rapid development of software.
- Produce system with a longer effective operational life
- Produce systems that most closely meet user needs and requirements.
- Produce system with excellent documentation
- Produce systems that needs less systems support
- Produce more flexible systems

Drawbacks of CASE

- Very Complex
- Expensive
- Difficult to customize
- Require training of maintenance staff
- Not easily maintainable
- Fragile(Weak)

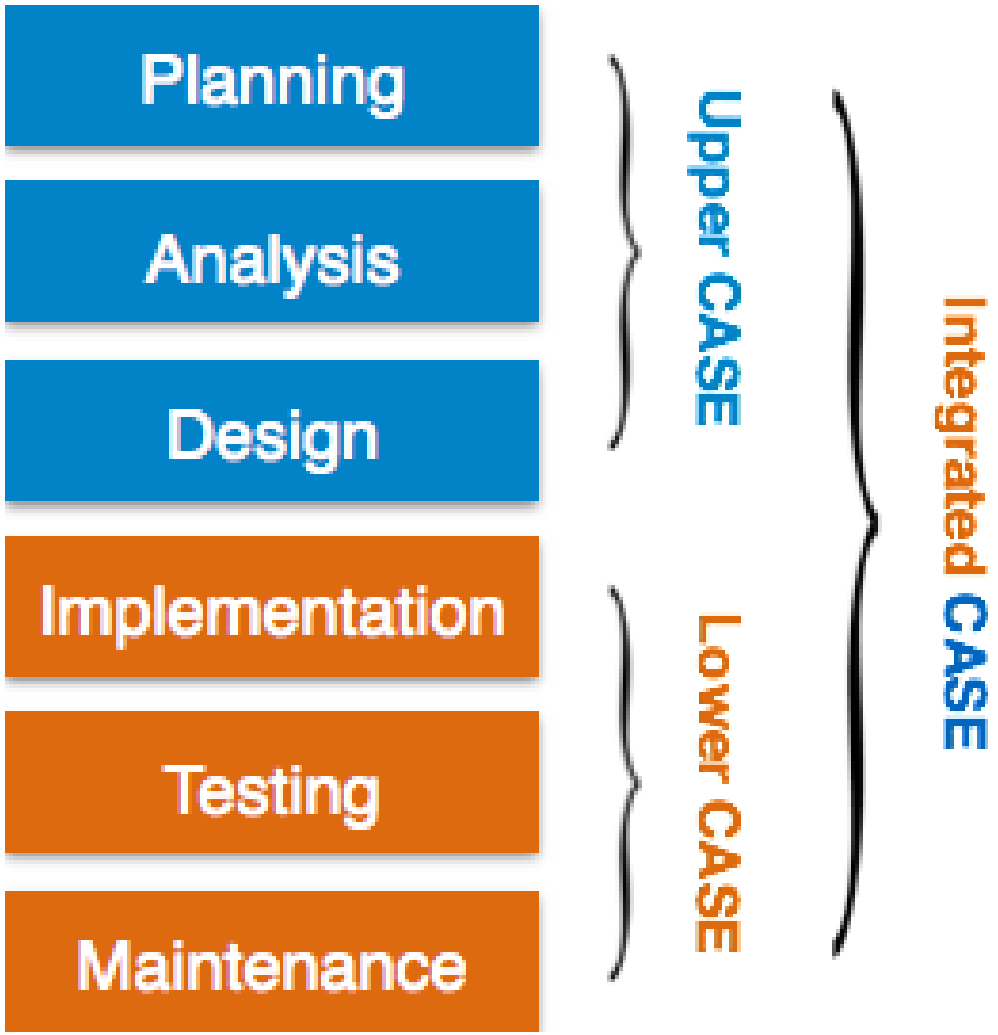
Components of CASE Tools



- Components of CASE Tools

- 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.


Planning
Analysis
Design
Implementation
Testing
Maintenance
Upper CASE
Lower CASE
Integrated CASE

Two types of tools used by software engineers

1. Analytical tools
   - Stepwise refinement
   - Cost-benefit analysis
   - Software metrics

2. CASE tools

- Case Tools Types
  - Diagram tools
  - Process Modeling Tools
  - Project Management Tools
  - Documentation Tools
  - Analysis Tools
  - Design Tools
  - Configuration Management Tools
  - Change Control Tools
  - Programming Tools
  - Prototyping Tools

- Web Development Tools
- Quality Assurance Tools
- Maintenance Tools

- Diagram tools
  - These tools are used to represent system components, data and control flow among various software components and system structure in a graphical form. For example, Flow Chart Maker tool for creating state-of-the-art flowcharts
  - Data Flow Diagrams (DFD)
  - Functional Hierarchy Diagrams
  - Entity-Relationship Diagrams
- Process Modeling Tools
  - Process modeling is method to create software process model, which is used to develop the software. Process modeling tools help the managers to choose a process model or modify it as per the requirement of software product. For example, EPF Composer

- Project Management Tools
  - These tools are used for project planning, cost and effort estimation, project scheduling and resource planning. Managers have to strictly comply project execution with every mentioned step in software project management. Project management tools help in storing and sharing project information in real-time throughout the organization. For example, Creative Pro Office, Trac Project, Basecamp.

- Documentation Tools
  - Documentation in a software project starts prior to the software process, goes throughout all phases of SDLC and after the completion of the project.
  - Documentation tools generate documents for technical users and end users. Technical users are mostly in-house professionals of the development team who refer to system manual, reference manual, training manual, installation manuals etc. The end user documents describe the functioning and how-to of the system such as user manual. For example, Doxygen, DrExplain, Adobe RoboHelp for documentation.

- Analysis Tools
  - These tools help to gather requirements, automatically check for any inconsistency, inaccuracy in the diagrams, data redundancies or erroneous omissions. For example, Accept 360, Accompa, CaseComplete for requirement analysis, Visible Analyst for total analysis.
- Design Tools
  - These tools help software designers to design the block structure of the software, which may further be broken down in smaller modules using refinement techniques. These tools provides detailing of each module and interconnections among modules. For example, Animated Software Design

- Configuration Management Tools
  - An instance of software is released under one version. Configuration Management tools deal with
    - Version and revision management
    - Baseline configuration management
    - Change control management
  - CASE tools help in this by automatic tracking, version management and release management. For example, Fossil, Git, Accu REV

- Programming Tools
  - These tools consist of programming environments like IDE (Integrated Development Environment), in-built modules library and simulation tools. These tools provide comprehensive aid in building software product and include features for simulation and testing. For example, Cscope to search code in C, Eclipse.
- Prototyping Tools
  - Software prototype is simulated version of the intended software product. Prototype provides initial look and feel of the product and simulates few aspect of actual product.

- Web Development Tools
  - These tools assist in designing web pages with all allied elements like forms, text, script, graphic and so on. Web tools also provide live preview of what is being developed and how will it look after completion. For example, Fontello, Adobe Edge Inspect, Foundation 3, Brackets
- Maintenance Tools
  - Software maintenance includes modifications in the software product after it is delivered. Automatic logging and error reporting techniques, automatic error ticket generation and root cause Analysis are few CASE tools, which help software organization in maintenance phase of SDLC. For example, Bugzilla for defect tracking, HP Quality Center.