***CHAPTER 2***

# PROGRAM DESIGN AND DEVELOPMENT

**Program/System design** is the process of defining the software structure (components/ modules, interfaces, and data) to satisfy/realize specified requirements.
**Program development** refers to all the activities and processes involved between the conception of the desired software through to the final manifestation of the software, in a planned and structured process.

## PROGRAM DEVELOPMENT CYCLE

This refers to the stages that form the framework for planning and controlling the creation of an information system. Several approaches to program development have been devised and the System Development Life Cycle (SDLC) is one of the most popular. The SDLC is a methodology that aims at producing a high quality system that **meets or exceeds customer expectations**, **reaches completion within times** and **cost estimates**, **works efficiently** and is **inexpensive to maintain and cost-effective to enhance**.

1) Project **planning, feasibility study:** The fundamental process of understanding why a system should be built and determining how the project team will go about building it. It should also establish a clear understanding of the current system. It involves
    a. Technical feasibility study: can the system be built
    b. Economic feasibility study: will the system provide business value, and what are the risks?
    c. Organizational feasibility study: if built, will it be used.
2) Systems **analysis, requirements definition:** the phase identifies the users of the system, what the system will do. It involves
    a. Analysis of the old system and ways to design the new system
    b. Requirement gathering. Various tools for collecting information are used. These include interviews, questionnaires, observation etc.
    c. Development of the new system proposal document.
3) Systems **design:** describes how the system will operate, in terms of hardware, software, network infrastructure, user interface, forms and reports that will be used, the specific programs, databases and files that will be needed. Design phase steps include;
    a. Design strategy: method of development, in-house, outsourced, or purchased
    b. Architecture design – hardware , software, internet infrastructure, and user interface
    c. Database and file specification
    d. Program design: defines the program that needs to be done and exactly what each will do.
4) System **Implementation:** The real code is written here.
5) **Integration and testing:** Brings all the pieces of the project together into a special testing environment, then checks for errors, bugs and interoperability.
6) **Acceptance, installation, deployment:** The final stage of initial development, where the software is put into use and runs actual business.
7) **Maintenance:** What happens during the rest of the software's life: changes, correction, additions, and moves to a different computing platforms etc. This step, perhaps most important of all, goes on seemingly forever.
   The SDLC is a cycle i.e. iterative in that a new requirement might initiate the whole process again.

# STRUCTURED PROGRAMMING DESIGN CONCEPTS

## 1. TOP-DOWN DESIGN

A **top-down** approach (also known as **stepwise design** or **deductive reasoning**, and in many cases used as a synonym of analysis or decomposition) is essentially the breaking down of a system to gain insight into its compositional sub-systems. In a top-down approach an overview of the system is formulated, specifying but not detailing any first-level subsystems. Each subsystem is then refined in yet greater detail, sometimes in many additional subsystem levels, until the entire specification is reduced to base elements. Top- down approach starts with the big picture. It breaks down from there into smaller segments.

Top-down design (also called " **Modular programming** " and "**stepwise refinement**") therefore, is a software design technique that emphasizes separating the functionality of a program into independent modules such that each module is designed to execute only one aspect of the desired functionality.

## ADVANTAGES OF MODULAR PROGRAMMING

- Allows a problem to be split in stages and for a team of programmers to work on the different modules thereby reducing program development time.
- Program modification and debugging is easier since changes can be isolated to specific modules.
- Modules can be tested and debugged independently.
- Since a module performs a specific and well defined task, it is possible to develop and place commonly used modules in a user library so that they can be accessed by different programs. This is also called code reuse. (E.g. Validation, Uppercase, Text color etc.)
- If a programmer cannot continue through the entire project, it is easier for another programmer to continue working on self-contained modules.



## 2. BOTTOM-UP DESIGN

A **bottom-up** approach (also known as inductive reasoning, and in many cases used as a synonym of synthesis) is the piecing together of systems to give rise to larger systems, thus making the original systems sub-systems of the emergent system. In a bottom-up approach the individual base elements of the system are first specified in great detail. These elements are then linked together to form larger subsystems, which then in turn are linked, sometimes in many levels, until a complete top-level system is formed.

**With this approach, there is more user and business awareness of the product**. Benefits are also realized in the early phases of development.

## 3. MONOLITHIC DESIGN

The monolithic design philosophy is that the application is responsible not just for a particular task, but can perform every step needed to complete a particular function

A monolithic application describes a software application which is designed without modularity.

# PROGRAM DESIGN TOOLS

## 1. Pseudo-code

Pseudo-code is an informal high-level description of the operating principle of a computer program or other algorithm and is intended for human reading rather than machine reading. Pseudo-code is a way to describe the algorithm in order to transform the algorithm into real source code.

For example, the Pseudo-code for comparing three numbers might be written: