2.1 Software Evolution

The s/w evolution has had distinct phases or layers of growth. These layers were built up one by one. With each layer representing an improvement over the previous one. Fig. (2.1) had shown layers of s/w.

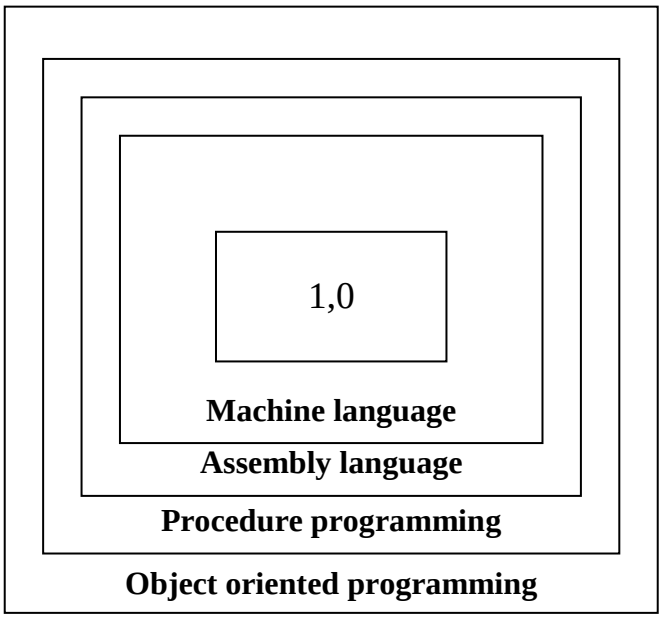

Fig. (2.1) layers of s/w technology

Since the invention of the computer, many programming approaches have been tried. These included techniques such as **modular programming**, **top-down programming, bottom-up programming and structured programming**. The primary motivation in each case has been the concern to handle the increasing complexity of programs that are reliable & maintainable. These techniques became popular among

programmers over the last two decades. With the advent of languages such as C, structured programming become very popular and was the main technique of the 1980s. Structured programming was a powerful tool that enabled programmers to write moderately complex programs fairly easily.

2.2 Procedure–Oriented Programming

Conventional programming using high level languages (COBOL, FORTRAN and C) is commonly known as procedure oriented programming. In the procedure-oriented approach, the problem is viewed as a sequence of things to be done, such as reading, calculating and printing. A number of functions are written to accomplish these tasks. A typical program structure for procedure programming is show in fig. (2.2).

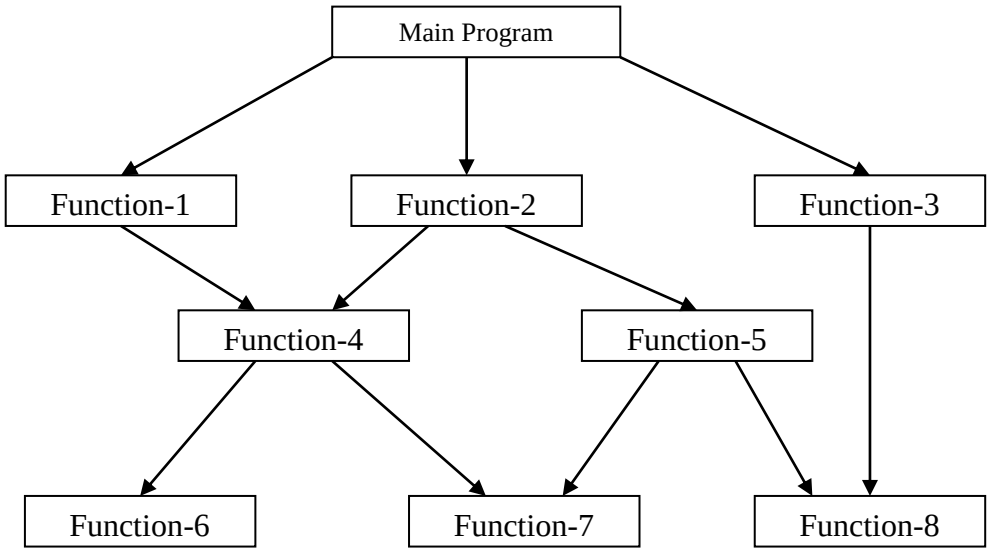

Fig. (2.2) Typical structure of procedure oriented programs

Procedure-oriented programming basically consists of writing a list of instructions for the computer to follow, and organizing these instructions into groups known as functions. We normally use a flowchart to organize these actions and represent the flow of control from one action to another.

In multi-function program, many important data items are placed as global so that they may be accessed by all the functions. Each function may have its own local data.

Disadvantage of Global data

- ❖ Global data are more vulnerable to an inadvertent change by a function.
- ❖ In a large- program it is very difficult to identify what data is used by which function.
- ❖ When we need to revise an external data structure, we should also revise all function that accesses the data.

Drawback of procedure approach is that it does not model real world problems very well. Because the functions are action-oriented and do not really correspond to the elements of the problem.

Characteristics exhibited by procedure-oriented programming

- ❖ Emphasis is on doing things (algorithms).
- ❖ Large programs are divided into smaller programs known as “functions”.
- ❖ Most of the functions share global data.
- ❖ Data move openly around the system from function to function.
- ❖ Function transforms data from one form to another.
- ❖ Employs top-down approach in program design.

2.3 Object-Oriented Programming Paradigm (OOP)

The major motivating factor in the invention of OOP is to salvage some of the flaws encountered the procedural approach. OOP treats data a critical element in the program development and does not allow it to flow freely around the system. It ties data more closely to the functions that operate on it and protects it from accidental modification from outside functions.

OOP allow us to decompose a problem in to a number of entities call objects and then built data and functions around these entities. The organization of data and functions in OOP is shown in Fig. (2.3).

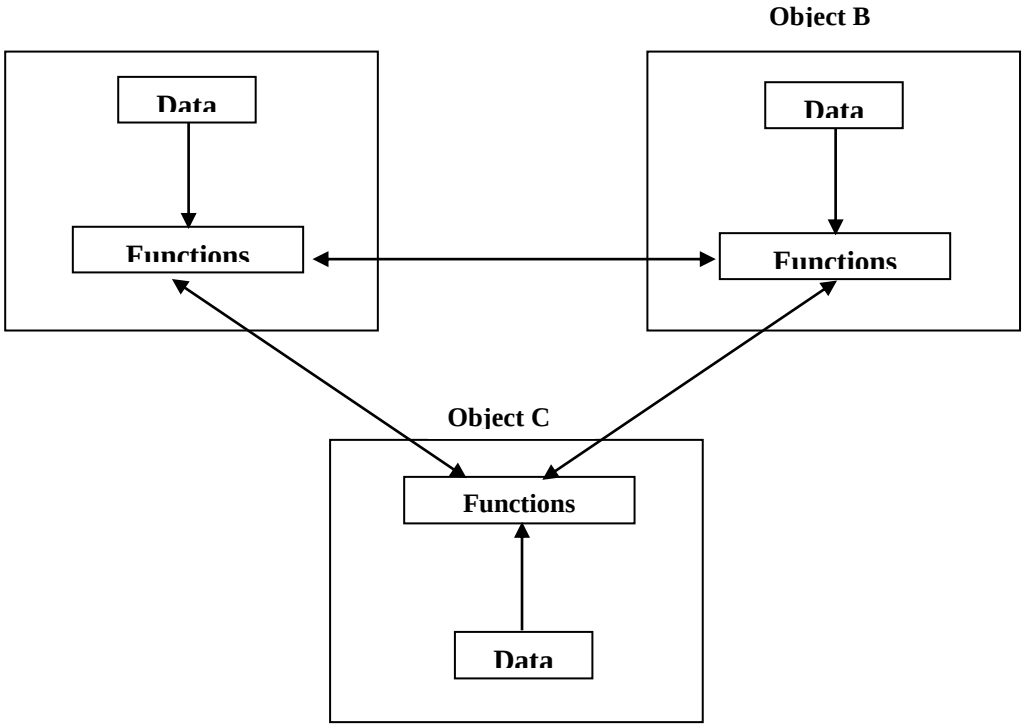

Fig. (2.3) organization of data and function in OOP

OOP: - Is approach that provides away of modularizing programs by creating partitioned memory area for both data and functions that can be used as templates for creating copies of such modules on demand.

The object-oriented approach to programming is an easy way to master the management and complexity in developing software systems that take advantage of the strengths of data abstraction.The data of an object can be accessed by the functions associated with that object. However, functions of one object can access the functions of other objects. Object-oriented programming (OOP) is a programming paradigm based on the

concept of objects, which are data structures that contain data, in the form of fields (or attributes) and code, in the form of procedures, (or methods). A distinguishing feature of objects is that an object's procedures provide access to and modify its fields.

Object:This is the basic unit of object-oriented programming. That is both data and function that operate on data are bundled as a unit called an object.

In object-oriented programming, computer programs are designed by making them out of objects that interact with one another. There is significant diversity in object-oriented programming, but most popular languages are class-based, meaning that objects are instances of classes, which typically also determines their type. Object orientation is an outgrowth of procedural programming. Procedural programming is a programming paradigm, derived from structured programming, based upon the concept of the procedure call. Procedures, also known as routines, subroutines, or methods define the computational steps to be carried out. Any given procedure might be called at any point during a program's execution, including by other procedures or itself. Procedural programming is a list or set of instructions telling a computer what to do step by step and how to perform from the first code to the second

code. Procedural programming languages include C, Fortran, Pascal, and BASIC.

The focus of procedural programming is to break down a programming task into a collection of variables, data structures, and subroutines, whereas in object-oriented programming it is to break down a programming task into objects that expose behavior (methods) and data (fields) using interfaces. The most important distinction is that while procedural programming uses procedures to operate on data structures, object-oriented programming bundles the two together, so an object, which is an instance of a class, operates on its "own" data structure.

2.4 Features of OOP: -

- ❖ Emphasis is on data rather than procedure.
  التركيز على البيانات وليس الإجراءات
- ❖ Programs are divided into objects.
  تنقسم البرامج إلى كائنات
- ❖ Data structures are designed such that they characterize the objects.
  تم تصميم هياكل البيانات بحيث تميز الأشياء
- ❖ Functions that operate on the data of an object are tied together in the data structure.
  ترتبط الوظائف التي تعمل على بيانات كائن معًا في بنية البيانات
- ❖ Data is hidden and cannot be accessed by external functions.

البيانات مخفية ولا يمكن الوصول إليها بوظائف خارجية

- Object may communicate with each other through functions.

  قد يتواصل الكائن مع بعضه البعض من خلال الوظائف

- New data and function can be easily added whenever necessary.

  يمكن إضافة بيانات ووظائف جديدة بسهولة عند الضرورة

- Follows bottom-up approach in program design.

  يتبع النهج التصاعدي في تصميم البرنامج

Benefits of OOP

OOP offers several benefits to both the program designer and the user. Object-orientation contributes to the solution of many problems associated with the development and quality of s/w products. The principal advantages are:

1. Through inheritance, we can eliminate redundant code and extend the use of existing classes.
2. We can build programs from the standard working modules that communicate with one another. This leads to saving of development time and higher productivity.
3. The principle of data hiding helps the programmer to build secure programs that cannot be by code in other parts of program.
4. It is possible to have multiple instances of an object to co-exist without any interference.

5. It is possible to map object in the problem domain to those objects in the program.
6. It is easy to partition the work in a project base on objects.
7. Object-oriented systems can be easily upgraded from small to large systems.
8. Massage passing techniques for communication between objects makes the interface descriptions with external systems much simpler.
9. S/w complexity can be easily managed.

Procedure oriented programming	**Object oriented programming**
Emphasis is on doing things (algorithms). **Logical structure**	Emphasis is on data rather than procedure.
Large programs are divided into smaller programs known as "functions". **Execution as functions**	Programs are divided into objects.
	Data structures are designed such that they characterize the objects.
	Functions that operate on the data of an object are tied together in the data structure.
Most of the functions share global data. **Focus on code**	Data is hidden and cannot be accessed by external functions.
Data move openly around the system from function to function. **Less data security**	Object may communicate with each other through functions.

Function transforms data from one form to another.	New data and function can be easily added whenever necessary.
Employs top-down approach in program design.	Follows bottom-up approach in program design.