**Software design startegies**

1. **Function oriented design**
2. **Object Oriented design**

**Function-oriented design**

The following are the salient features of a typical function-oriented design approach:

1. A system is viewed as something that performs a set of functions. Starting at this high-level view of the system, each function is successively refined into more detailed functions. For example, consider a function create-newlibrary-member which essentially creates the record for a new member, assigns a unique membership number to him, and prints a bill towards his membership charge.
2. This function may consist of the following subfunctions:

• assign-membership-number

• create-member-record

• print-bill

Each of these sub-functions may be split into more detailed sub functions and so on.

2. The system state is centralized and shared among different functions, e.g. data such as member-records is available for reference and updation to several functions such as:

create-new-member

• delete-member

• update-member-record

**Object-oriented design**

In the object-oriented design approach, the system is viewed as collection of objects (i.e. entities). The state is decentralized among the objects and each object manages its own state information. For example, in a Library Automation Software, each library member may be a separate object with its own data and functions to operate on these data. In fact, the functions defined for one object cannot refer or change data of other objects. Objects have their own internal data which define their state. Similar objects constitute a class. In other words, each object is a member of some class. Classes may inherit features from super class. Conceptually, objects communicate by message passing.

**Function-oriented vs. object-oriented design approach**

The following are some of the important differences between function-oriented and object-oriented design.

 • Unlike function-oriented design methods, in OOD, the basic abstraction are not real-world functions such as sort, display, track, etc, but realworld entities such as employee, picture, machine, radar system, etc. For example in OOD, an employee pay-roll software is not developed by designing functions such as update-employee-record, getemployee-address, etc. but by designing objects such as employees, departments, etc. Grady Booch sums up this difference as “identify verbs if you are after procedural design and nouns if you are after object-oriented design”

* In OOD, state information is not represented in a centralized shared memory but is distributed among the objects of the system. For example, while developing an employee pay-roll system, the employee data such as the names of the employees, their code numbers, basic salaries, etc. are usually implemented as global data in a traditional programming system; whereas in an object-oriented system these data are distributed among different employee objects of the system. Objects communicate by message passing. Therefore, one object may discover the state information of another object by interrogating it. Of course, somewhere or other the real-world functions must be implemented. In OOD, the functions are usually associated with specific real-world entities (objects); they directly access only part of the system state information.
* Function-oriented techniques such as SA/SD group functions together if, as a group, they constitute a higher-level function. On the other hand, object-oriented techniques group functions together on the basis of the data they operate on. To illustrate the differences between the object-oriented and the function-oriented design approaches, an example can be considered.

Example: Fire-Alarm System

 The owner of a large multi-stored building wants to have a computerized fire alarm system for his building. Smoke detectors and fire alarms would be placed in each room of the building. The fire alarm system would monitor the status of these smoke detectors. Whenever a fire condition is reported by any of the smoke detectors, the fire alarm system should determine the location at which the fire condition is reported by any of the smoke detectors, the fire alarm system should determine the location at which the fire condition has occurred and then sound the alarms only in the neighboring locations. The fire alarm system should also flash an alarm message on the computer console. Fire fighting personnel man the console round the clock. After a fire condition has been successfully handled, the fire alarm system should support resetting the alarms by the fire fighting personnel.

Function-Oriented Approach:

/\* Global data (system state) accessible by various functions \*/

BOOL detector\_status[MAX\_ROOMS];

int detector\_locs[MAX\_ROOMS];

BOOL alarm\_status[MAX\_ROOMS]; /\* alarm activated when status is set \*/

 int alarm\_locs[MAX\_ROOMS]; /\* room number where alarm is located \*/

 int neighbor-alarm[MAX\_ROOMS][10]; /\* each detector has at most 10 neighboring locations \*/

 The functions which operate on the system state are:

 interrogate\_detectors();

get\_detector\_location();

determine\_neighbor();

 ring\_alarm();

 reset\_alarm();

report\_fire\_location();

Object-Oriented Approach:

class detector

 attributes: status, location, neighbors

operations: create, sense\_status, get\_location, find\_neighbors

class alarm

attributes: location, status

operations: create, ring\_alarm, get\_location, reset\_alarm

In the object oriented program, an appropriate number of instances of the class detector and alarm should be created. If the function-oriented and the object oriented programs are examined, it can be seen that in the function-oriented program, the system state is centralized and several functions accessing this central data are defined. In case of the object-oriented program, the state information is distributed among various sensor and alarm objects.

It is not necessary an object-oriented design be implemented by using an object-oriented language only. However, an object-oriented language such as C++ supports the definition of all the basic mechanisms of class, inheritance, objects, methods, etc. and also support all key object-oriented concepts that we have just discussed. Thus, an object-oriented language facilitates the implementation of an OOD. However, an OOD can as well be implemented using a conventional procedural language – though it may require more effort to implement an OOD using a procedural language as compared to the effort required for implementing the same design using an object-oriented language.

Even though object-oriented and function-oriented approaches are remarkably different approaches to software design, yet they do not replace each other but complement each other in some sense. For example, usually one applies the top-down function-oriented techniques to design the internal methods of a class, once the classes are identified. In this case, though outwardly the system appears to have been developed in an object-oriented fashion, but inside each class there may be a small hierarchy of functions designed in a top-down manner.