Solved Question Paper

June 2016

1.b)Explain the role of message passing in object oriented system.(5marks)

Objects are usually components of a larger program or system. Message passing is a type of communication between processes, objects or other resources. Software objects interact and communicate with each other by message passing. Through this communication the functionalities are achieved. This is the process where an object sends data to another object or asks the object to invoke a method. When object X wants object Y to perform a method of object Y, object X sends a message to object Y.

Benefits of message passing

- An object's characteristics are expressed through its methods, so message passing supports all possible interactions between objects.
- It closes gap between objects. Objects do not need to be in the same process, or even on the same machine, to send and receive messages back and forth to each other.

Message passing can be **synchronous** or **asynchronous**. Synchronous message passing systems require the sender and receiver to wait for each other while transferring the message. In asynchronous communication the sender and receiver do not wait for each other and can carry on their own computations while transfer of messages is being done.

Three steps for message passing are :

1. Creating classes that define objects and behaviour.
2. Creating objects from class definition
3. Establishing communication among objects.

4.a)What do you understand by the term "Concurrency" ? Which model is perfect enough to describe concurrency in a UML diagram and why ? When can you say that two objects are concurrent ? Briefly describe any two concurrency issues.(10marks)

Concurrency is an essential concept in computer programming. It is the idea of handling more than one task at a time.

Many programming problems require that the program be able to :

- Stop what it's doing
- Currently deal with some other problem, and
- Return to the main process. There is a large class of problems in which you have to partition the problem into separately running pieces so that the whole program can be more responsive. Within a program, these separately running pieces are called threads, and the general concept called multithreading.

If we have more than one thread running that is expecting to access the same resource then there is a problem. To avoid this problem, a thread locks a resource, completes its task, and then releases the lock so that someone else can use it at a time. It is very important to handle threads properly.

In terms of object, concurrency in objects can be identified by the way they change the state. Concurrent objects are those that can change state independently. Aggregation implies concurrency.

Concurrency in OOAD is described and studied in dynamic modelling.

One important issue in system design is to find the concurrency in objects. If we identify non-concurrent objects, we can assign all the objects together in one thread of control, or process.

If the objects are concurrent in nature we have to assign them to different thread of control.

A thread of control is a path through a set of state diagrams on which a single object is active at a time.

Objects are shared among threads, that is, several methods of the same object can be active at the same time.

Thread splitting : Object sends a message but does not wait for the completion of the method.

Identification of concurrency :

Concurrency is identified in a dynamic model. Two objects are said to be concurrent (parallel) if they can receive events at the same time. Concurrent objects are required to be assigned to different threads of control.

If an object must perform two or more activities concurrently, then the events must be synchronized.

Concurrency issues :

- Data integrity : Threads accessing the same object need to be synchronized, for example banking account.
- Deadlock : One or more threads in the system are permanently blocked. Example : Thread A waiting on Thread b, which is waiting on Thread A.
- Starvation : A thread is not getting enough resources to accomplish its work. Example : All requests from one user are being handled before another users requests.

4.c)How are constraints implemented in object oriented languages ? Give an example/code in support of your answer. (5marks)

Class constraints describe relationships that must be between the attribute values of an instance of the class; and preconditions and post-conditions specify what must be true before and after an operation is called. Once these are implemented by including code in the class which checks those conditions at the appropriate times, then the application becomes more reliable and robust.

All the preconditions that are specified for an operation should be explicitly checked in an implementation. Preconditions state properties of operation’s parameters that must be satisfied if the operation is to be able to run to completion successfully. It is the responsibility of the caller of the operation to ensure that the precondition is satisfied when an operation is called.

If an operation does not check its parameter values, then, there is a possibility that wrong or meaningless values will go undetected, and this results in unpredictable run-time errors. A better strategy is to check its precondition for an operation and to raise an exception if a violation of the detection is detected.

Example :

```
Public class SavingsAccount

{
```

```
Public void withdraw(double amt)

{

If( amt >=balance)

{

// throw PreConditionUnsatisfied

}

balance=balance-amt;

}

Private double balance;

}
```

Any constraints can be checked at run-time by writing code that will validate the current state of the model. These checks increase overhead. That is why, except for the case of precondition checking, constraints are rarely implemented explicitly.

5.a)What is object oriented decomposition of systems ? Explain briefly.(5marks)

Decomposition is an important technique for coping with complexity based on the idea of divide and conquer. Decomposition is dividing a problem into subproblems. By doing this the problem becomes less complex and easier to overlook and deal with.

Decomposition can be in two forms : process-oriented and object-oriented decomposition

Process-oriented decompositions divide a complex process, function or task into simpler sub processes until they are simple enough to be dealt with. The solutions of these subfunctions, then, need to be executed in certain sequential or parallel orders, in order to obtain a solution to the complex process.

Object oriented decomposition aims at identifying individual autonomous objects that encapsulate both a state and a certain behaviour. Each major components of the system is called a subsystem. Then communication among these objects leads to the desired solutions.

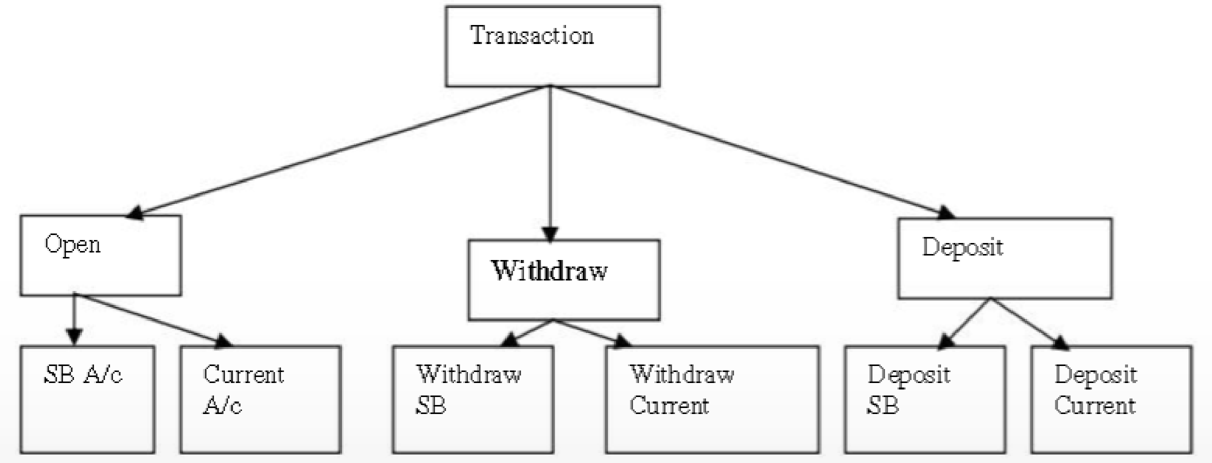

Figure 3: A Function data decomposition (SAD)

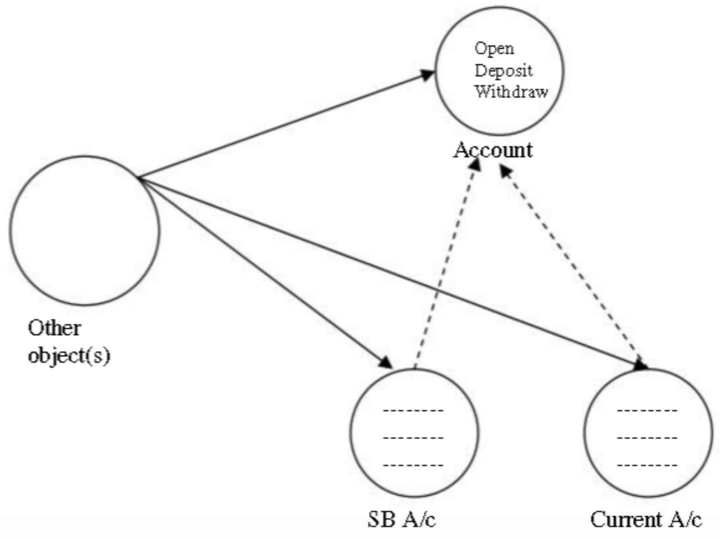

Figure 4: OO Decomposition (OOD)

Object-oriented solution is a favourable solution because the object-oriented approach provides a semantically richer framework that leads to decompositions that are more closely related to entities from the real world. The identification of abstractions supports more abstract solutions to be reused, and the object-oriented approach supports the evolution of systems better, as those concepts that are more likely to be changed can be hidden within the objects.

Object oriented decomposition of systems tend to be better able to cope with change. Each subsystem has a well-defined interface that communicate with rest of the system. Each of these interfaces defines all form of interaction that are required for proper functioning of he system, but the internal implementations are left to the subsystem itself. This is because they manage to bind those items that are likely to change within an object and hide them from the outside world. The advantage of this is that, the outside cannot see them, and therefor be dependent on them and does not need to be changed of these items change. Object-oriented decompositions are closer to the problems domain, as they directly represent the real-world entities in their structure and behaviour. Object orientation has the advantage of continuity throughout analysis, design implementation, and persistent representation.