

Study questions for Lecture 10 - Answers

1. What is the difference between a transition in a flow diagram and one in a state chart, in terms of its purpose?

In the flow diagram its purpose is to show flow of control to the next operation(s)/decisions. If there are multiple flows from a (decision) node, they have conditions on the decision outcome to show which flow is correct. There is no idea of stopping at a node and waiting for something, once a node action or decision is finished, transition to the next node takes place.

In the state diagram, they show transitions that occur in response to events that are caused by agents outside the modeled system. If there are conditions, they too must be satisfied if the transition is to take place, but the event is what causes the transition. If there are no events, no transition takes place.

2. In the concurrent state sample diagram, how many possible system states are there?

There were two subsystems with two states each. This means there can be a total four combinations, each a possible state in the compound system.

3. Consider, for each pattern: when is the choice of alternatives determined, and when can it be set or changed. (Times: class design time, object creation time, post object creation run)

Strategy pattern: choice of alternative classes can be made when the program is written, or later. You can later create more alternative strategy classes, as long as they subclass the required interface. The actual alternative object is created when the strategy variable is set, often at start up time, but it could be reset during run time.

Decorator pattern: alternative class created/chosen when the program is designed. Actual alternative object created during run time (i.e. the instance of the decorator object).

State pattern: Possible alternative classes decided during program design. Actual alternative state object is created at run time. In the book they "pre-create" a set of substate objects when the host object is constructed. They could also be created each time a state change occurs, but this is less object-efficient.

Factory pattern. The possible alternatives are unknown when the original program is created. During reuse of a framework, you will specify the alternative classes with class definitions that subclass/extend the interfaces in the original factory design framework. The actual alternative objects will be automatically created during run time when instances of the subclass are created.

Command pattern: The possible alternatives classes are created during program design time. The instance objects are created during run time, when an instance of the current subclass is created and assigned to the command variable.