UNIT-V

Behavioralpattern

Insoftwareengineering,behavioraldesignpatternsaredesignpatternsthatidentifycommoncommunication patterns between objects and realize these patterns. By doing so, these patterns increase flexibility in carrying out thiscommunication.

Examples of this type of design pattern include:

- Chain of responsibility pattern: Command objects are handled or passed on to other objects by logic- containing processing objects
- Command pattern: Command objects encapsulate an action and itsparameters
- "Externalize the Stack": Turn a recursive function into an iterative one that uses astack.[1]
- Interpreter pattern: Implement a specialized computer language to rapidly solve a specific set ofproblems
- Iterator pattern: Iterators are used to access the elements of an aggregate object sequentially without exposing its underlyingrepresentation
- Mediator pattern: Provides a unified interface to a set of interfaces in asubsystem
- Memento pattern: Provides the ability to restore an object to its previous state(rollback)
- Observer pattern: aka Publish/Subscribe or Event Listener. Objects register to observe an event that may be raised by anotherobject
  - Weak reference pattern: De-couple an observer from anobservable.[2]
- State pattern: A clean way for an object to partially change its type atruntime
- Strategy pattern: Algorithms can be selected on thefly
- Template method pattern: Describes the program skeleton of aprogram
- Visitor pattern: A way to separate an algorithm from anobject

InObjectOrientedDesign,thechain-of-responsibilitypatternisadesignpatternconsistingofa sourceofcommandobjectsandaseriesofprocessingobjects.Eachprocessingobjectcontains logic that defines the types of command objects that it can handle; the rest are passed to thenextprocessing object in the chain. A mechanism also exists for adding new processing objects to the end of this chain.In object-oriented programming, the command pattern is a design pattern in which an object is used to represent and encapsulate all the information needed to call a method at a later time. This information includes the method name, the object that owns the method and values for the method parameters.

Incomputerprogramming,theinterpreterpatternisadesignpatternthatspecifieshowtoevaluate

sentences in a language. The basic idea is to have a class for each symbol (terminal or nonterminal) in a specialized computer language. The syntax tree of a sentence in the language is an instance of the composite pattern and is used to evaluate (interpret) thesentence.

Inobject-orientedprogramming,theiteratorpatternisadesignpatterninwhichaniteratorisused to traverse a container and access the container's elements. The iteratorpattern decouples algorithms from containers; in some cases, algorithms are necessarily container-specific and thus cannot be decoupled.

Themediatorpatterndefinesanobjectthatencapsulateshowasetofobjectsinteract.Thispattern is considered to be a behavioral patterndue to the way it can alter the program's runningbehavior.

Usually a program is made up of a (sometimes large) number of classes. So the logicand computationis distributed among these classes. However, as more classes are developed in a program, especially during maintenanceand/or refactoring, the problem of communicationbetween these classes may become more complex. This makes the program harder to read and maintain. Furthermore, it can become difficult to change the program, since any change may affect code in several other classes.

Withthemediatorpattern,communicationbetweenobjectsisencapsulatedwithamediatorobject. Objects no longer communicate directly with each other, but instead communicate through the mediator. This reduces the dependencies between communicating objects, therebylowering the coupling

.

Thememento patternisasoftwaredesignpatternthatprovidestheabilitytorestoreanobjecttoits previous state (undoviarollback).

The memento pattern is implemented with two objects: the *originator* and a *caretaker*. The originator is some object that has an internal state. The caretaker is going to do something to the originator, but wants to be able to undo the change. The caretaker first asks the originator for a memento object. Then it does whatever operation (or sequence of operations) it was going to do. To roll back to the state before the operations, it returns the memento object to the originator. The memento object itself is anopaque object(one which the caretaker cannot, or should not, change). When using this pattern, care should be taken if the originator may change other objects or resources - the memento pattern operates on a singleobject.

Theobserver patternisasoftwaredesignpatterninwhichanobject,calledthesubject,maintains a list of its dependents, called observers, and notifies them automatically of any state changes, usually by calling one of their methods. It is mainly used to implement distributedevent handling systems. Observer is also a key part in the familiar MVCarchitectural pattern. In fact the observer pattern was first implemented in Smalltalk's MVC based user interface framework.The state pattern, which closely resembles Strategy Pattern, is a behavioral software designpattern, also known as the objects for states pattern. This pattern is used incomputerprogramming to represent the state of an object. This is a clean way for an object to partially changeits type at runtime

In computer programming, the strategy pattern (also known as the policy pattern) is a particular software design pattern, whereby algorithms can be selected at runtime. Formally speaking, the strategy pattern defines a family of algorithms, encapsulates each one, and makes them interchangeable. Strategy lets the algorithm vary independently from clients that use it.

A *template method* defines the program skeletonof an algorithm. One or more of the algorithm steps can be overridden by subclasses to allow differing behaviors while ensuring that the overarching algorithm is still followed.

In object-oriented programming, first a class is created that provides the basic steps of an algorithmdesign. These steps are implemented using abstract methods. Later on, subclasses change the abstract methods to implement real actions. Thus the general algorithm is saved in one place but the concrete steps may be changed by the subclasses.

Inobject-orientedprogrammingandsoftwareengineering,thevisitordesignpatternisawayof separating an algorithmfrom an object structure on which it operates. A practical result of this separation is the ability to add new operations to existing object structures without modifying those structures. It is one way to easily follow the open/closed principle.

Behavioral patterns				
Name	Description	In *Design Patterns*	In *Code Complete*[17]	Other
Blackboard	Generalized observer, which allows multiple readers and writers. Communicates information system- wide.	No	No	N/A
Chain of responsibility	Avoid coupling the sender of a request to its receiver by giving more than one object a chance to handle the request. Chain the receiving objects and pass the request along the chain until an object handles it.	Yes	No	N/A
Command	Encapsulate a request as an object, thereby letting you parameterize	Yes	No	N/A

	clients with different requests, queue or log requests, and support undoableoperations.			
Interpreter	Given a language, define a representation for its grammar along with an interpreter that uses the representation to interpret sentences in thelanguage.	Yes	No	N/A
Iterator	Provide a way to access the elements of an aggregateobject sequentially without exposing its underlying representation.		Yes	N/A
Mediator	Define an object that encapsulates how a set of objects interact. Mediator promotes loosecouplingby keeping objects from referring to each other explicitly, and it lets you vary their interaction independently.	Yes Yes	No	N/A
Memento	Without violating encapsulation, capture and externalize an object's internal state allowing the object to be restored to this state later.	Yes	No	N/A
Null object	Avoid null references by providing a default object.	No	No	N/A

Observer or Publish/subscribe	Define a one-to-many dependency between objects where a state change in one object results in all its dependents being notified and	Yes	Yes	N/A

	updated automatically.			
Servant	Define common functionality for a group of classes	No	No	N/A
Specification	Recombinable business logicin a Booleanfashion	No	No	N/A
State	Allow an object to alter its behavior when its internal state changes. The object will appear to change its class.	Yes	No	N/A
Strategy	Define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it.	Yes	Yes	N/A
Template method	Define the skeleton of an algorithm in an operation, deferring some steps to subclasses. Template method lets subclasses redefine certain steps of an algorithm without changing the algorithm's structure.	Yes	Yes	N/A
Visitor	Represent an operation to be performed on the elements of an object structure. Visitor lets you define a new operation without changing the classes of the elements on which it	Yes	No	N/A

	operates.			

Separate interface from implementation