State

Design Patterns

Behavioral Design Patterns

- Chain of responsibility
- Command
- Interpreter
- Iterator
- Mediator
- Memento
- Observer
- State
- Strategy
- Template method
- Visitor

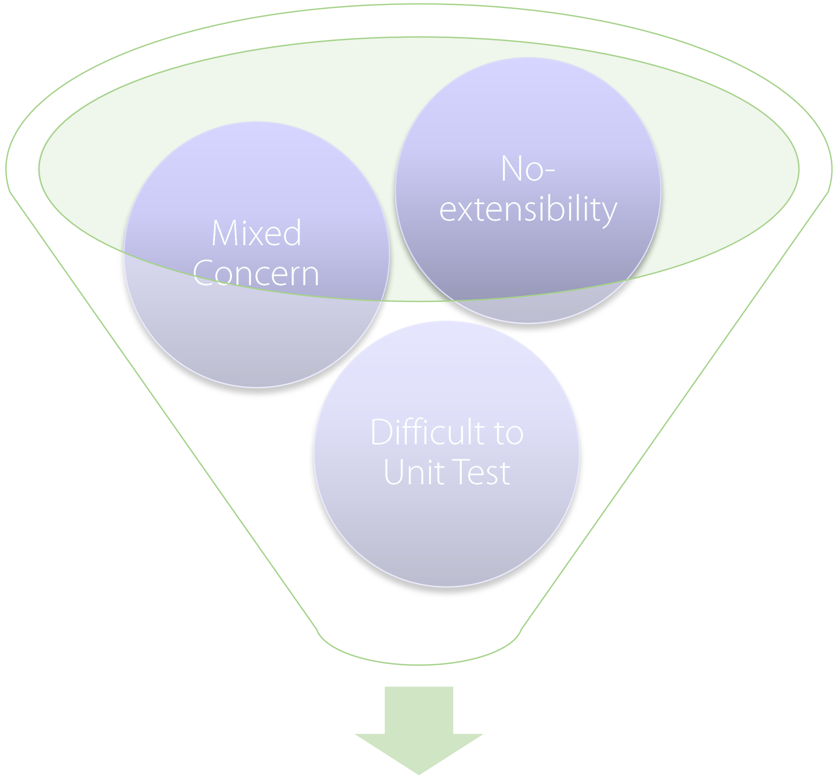
Motivating Example

Work Item Tracking	Features	CMMI	Agile
Bugzilla	Multiple States	Proposed	Active
Clear Quest	Different Behaviors	Active	Resolved
Team Foundation Server		Resolved	Closed
		Closed	

Simple Version Method Logic

```
public void Delete()
{
    switch (this.State)
    {
      case "Proposed":
        unitOfWork.Entities.Remove(this);
       break;
      case "Active":
        Console.WriteLine("Work Item is already active. Cannot Delete.");
        break;
      case "Resolved":
        Console.WriteLine("Work Item is already resolved. Cannot Delete.");
        break;
      case "Closed":
        unitOfWork.Entities.Remove(this);
        break;
    }
}
```

Issues with the Simple Approach

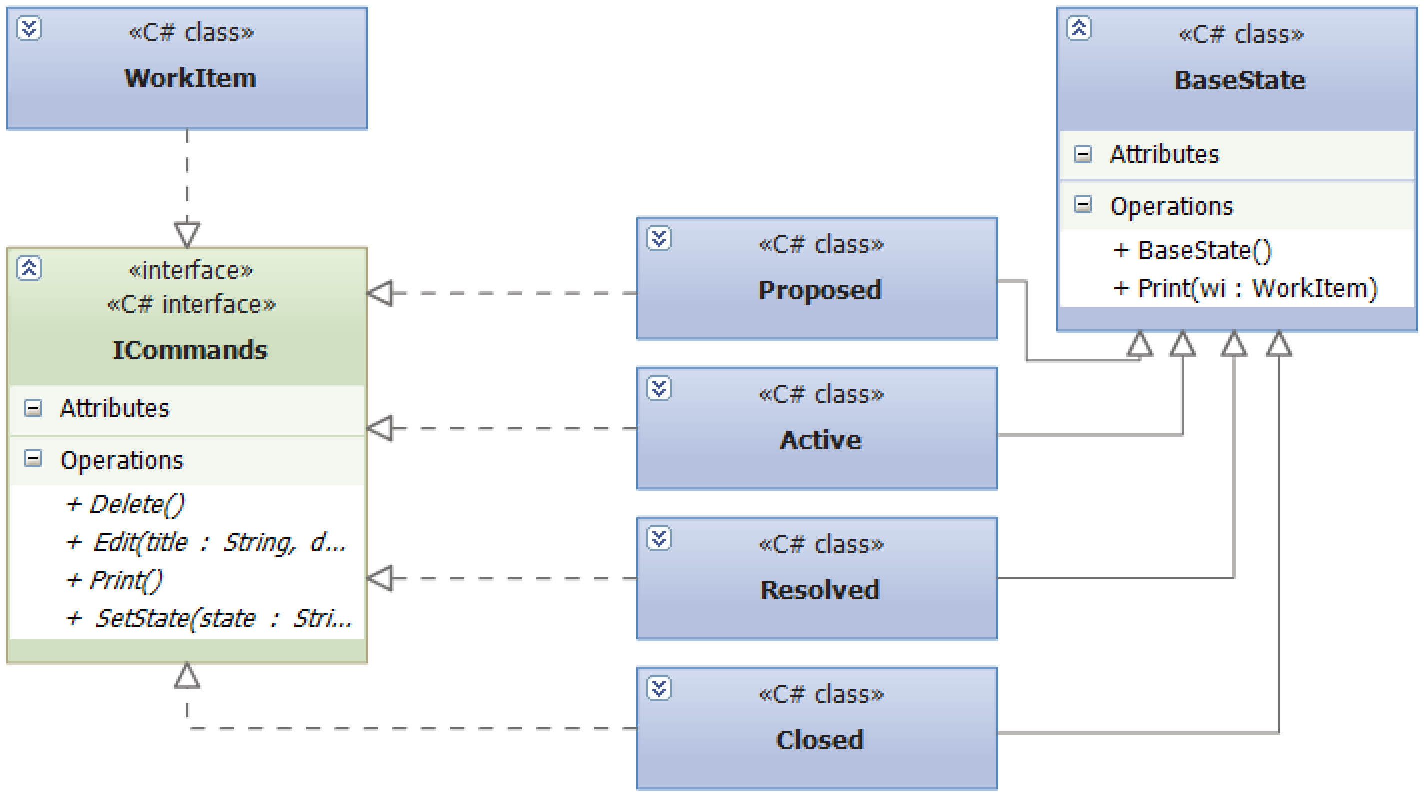


Maintenance Problem

Intent of the State Pattern

- Change behavior of the object with each state
- Encapsulate the logic of each state into a single object
- Allow for dynamic state discovery
- Make unit testing easier

Structure


«C# class»
WorkItem
«interface»
«C# interface»
ICommands
Attributes
Operations
+ Delete()
+ Edit(title : String, d...
+ Print()
+ SetState(state : Stri...
«C# class»
Proposed
«C# class»
Active
«C# class»
Resolved
«C# class»
Closed
«C# class»
BaseState
Attributes
Operations
+ BaseState()
+ Print(wi : WorkItem)

Common Command Interface

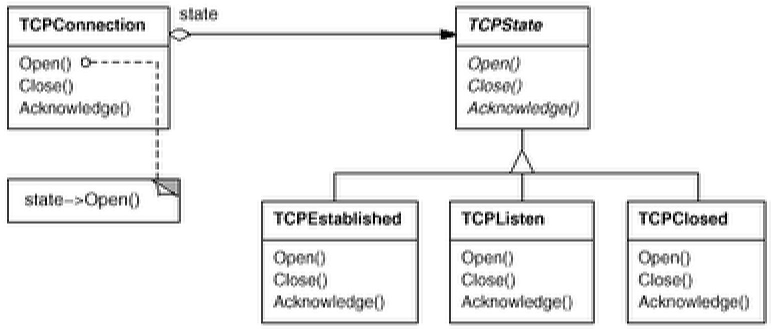
```
interface ICommands
{
   void Delete();
   void Edit(string title, string desc);
   void Print();
   void SetState(string state);
}
```

Benefits of State Pattern

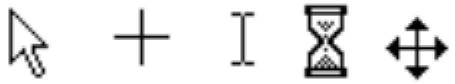
- Separation of Concerns
- Localization of state-specific behavior
- Transition between states is explicit and clear
- Reuse of the state objects
- Simplify the program
- Easier Maintainability

Known Uses

- TCP Connection Protocols



- Mouse Pointer objects during Drag & Drop



- Email POP Servers

Summary

- State Pattern is a Behavioral Pattern
- Use when the behavior requires a change at runtime
- Separate the concerns and divide the states into classes dedicated to one state
- Indicator to use the State Pattern is the proliferous use of a switch
- Can allow for truly dynamic states

For more in-depth online developer training visit

pluralsight
see what you can learn

on-demand content from authors you trust