

Mediator DP

1-Mar-23 5:15 PM

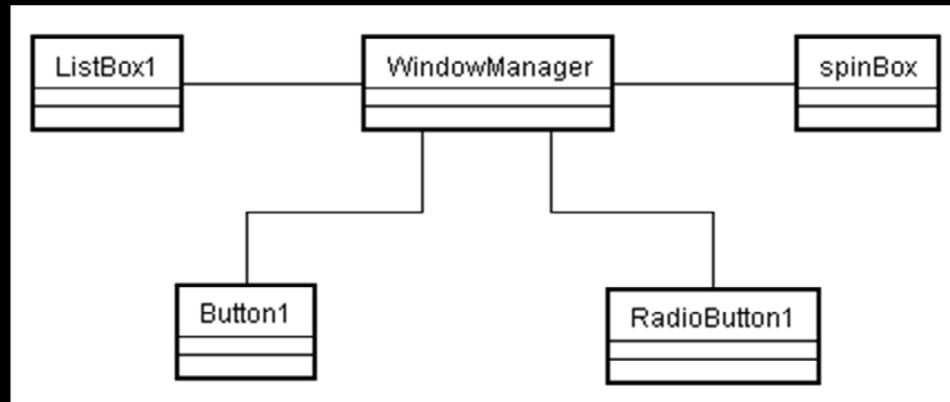
1

Problem

- We have a set of graphic components in a window: list box, button, radio buttons, spin box, etc.
 - A change in the value of one graphic components causes others to change.
 - How do we implement the design?



Solution



1-Mar-23 5:15 PM

3

Do not ask every graphic object to communicate with others.

Each graphic object communicates with a window manager. The window manager communicates with other GUI components.

It replaces many-to-many interactions with multiple one-to-many interactions.

One-to-many interactions are easier to understand, maintain and extend.

It centralizes control that otherwise would be distributed in many objects.

It is commonly used to coordinate related GUI components.

=====

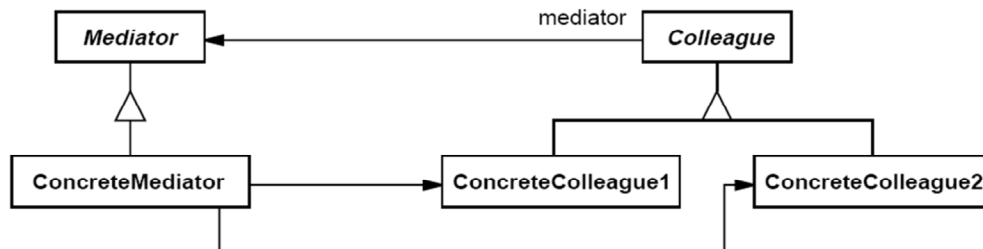
Defines an object that encapsulates how a set of objects interact.

A mediator is responsible for controlling and coordinating the interactions of a group of objects.

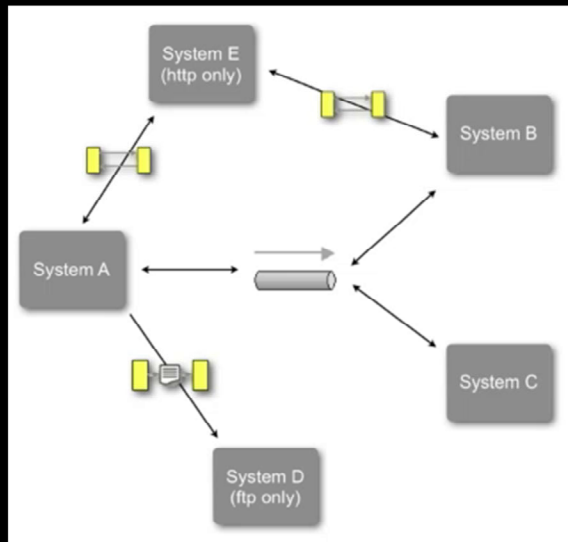
Mediator promotes loose coupling by preventing objects from referring to each other explicitly.

It lets us vary the interaction between objects independently.

Mediator Structure



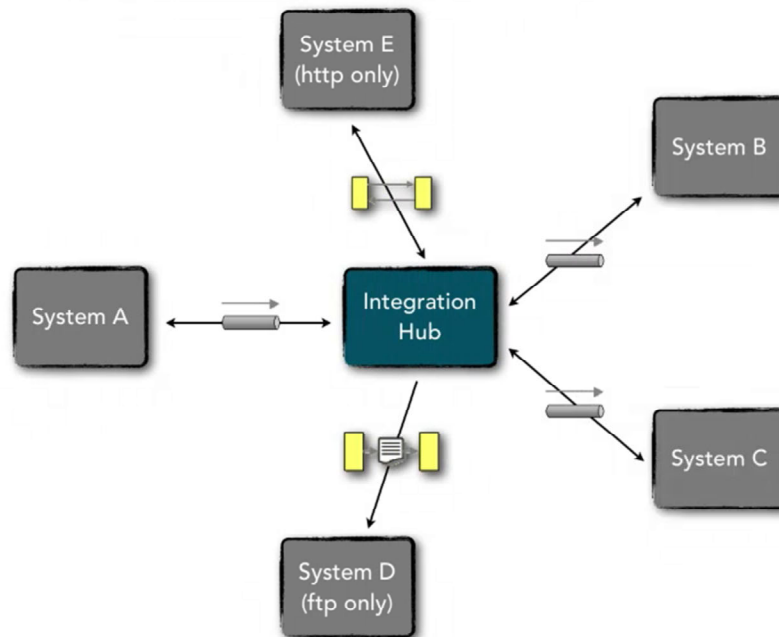
A complex, highly dependent spaghetti topology



1-Mar-23 5:15 PM

5

Integration Hubs



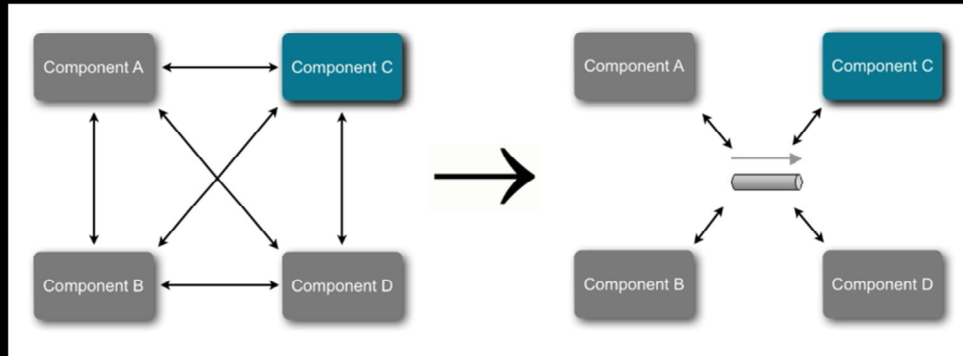
1-Mar-23 5:15 PM

6

Open source Integration Hubs: Mule, Camel

Proprietary Integration Hubs: Websphere ESB, etc.

Lower Coupling

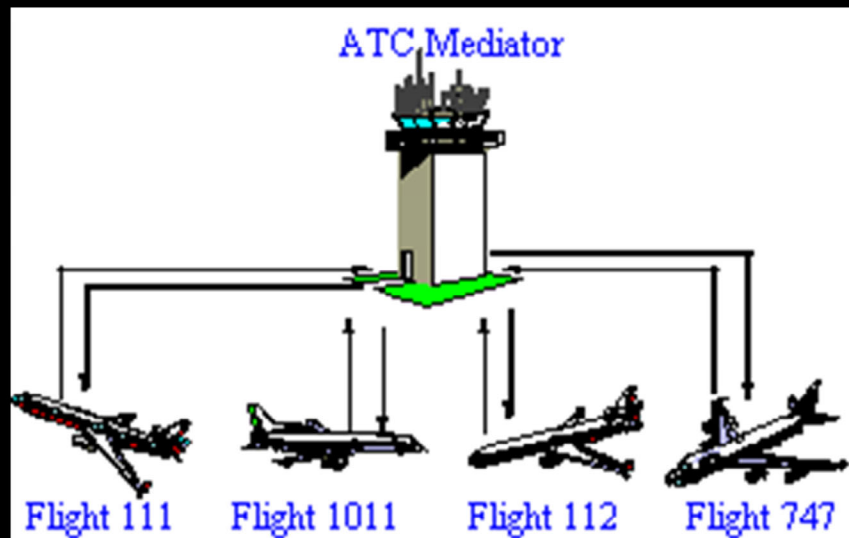


1-Mar-23 5:15 PM

7

Component can be a class, package, Jar file or some application.

Real Life Analog



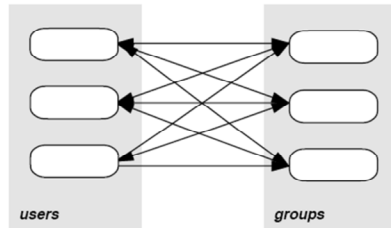
1-Mar-23 5:15 PM

8

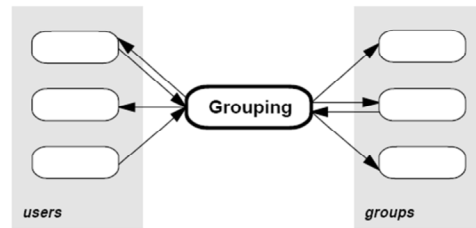
The *Mediator* defines an object that controls how a set of objects interact. Loose coupling between colleague objects is achieved by having colleagues communicate with the *Mediator*, rather than with each other. The control tower at a controlled airport demonstrates this pattern very well. The pilots of the planes approaching or departing the terminal area communicate with the tower, rather than explicitly communicating with one another. The constraints on who can take off or land are enforced by the tower. It is important to note that the tower does not control the whole flight. It exists only to enforce constraints in the terminal area.

Users and Groups on Linux

Before:



After:

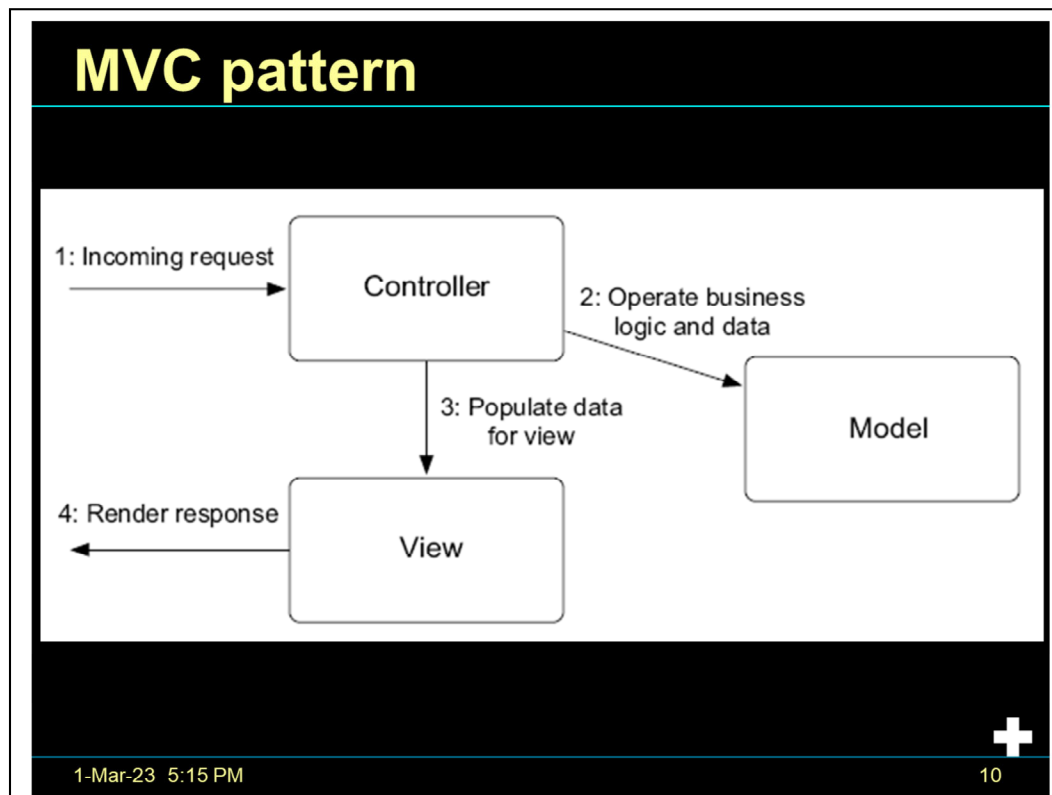


Would you use Composite here instead of Mediator?

1-Mar-23 5:15 PM

9

User and Groups is not the same on Linux. So Composite cannot be used.
Grouping may be a Singleton



Several problems can arise when applications contain a mixture of data access code, business domain logic code, and presentation code.

Such applications are difficult to maintain, because interdependencies between all of the components cause strong ripple effects whenever a change is made anywhere.

The Model-View-Controller design pattern solves these problems by decoupling business domain logic and data presentation.

MVC is an example of Mediator design pattern.

=====

Struts overview

Client browser

An HTTP request from the client browser creates an event. The Web container will respond with an HTTP response.

Controller

The Controller receives the request from the browser, and makes the decision where to send the request. With Struts, the Controller is a command design pattern implemented as a servlet. The struts-config.xml file configures the Controller.

Business logic

The business logic updates the state of the model and helps control the flow of the application. With Struts this is done with an Action class as a thin wrapper to the

actual business logic.

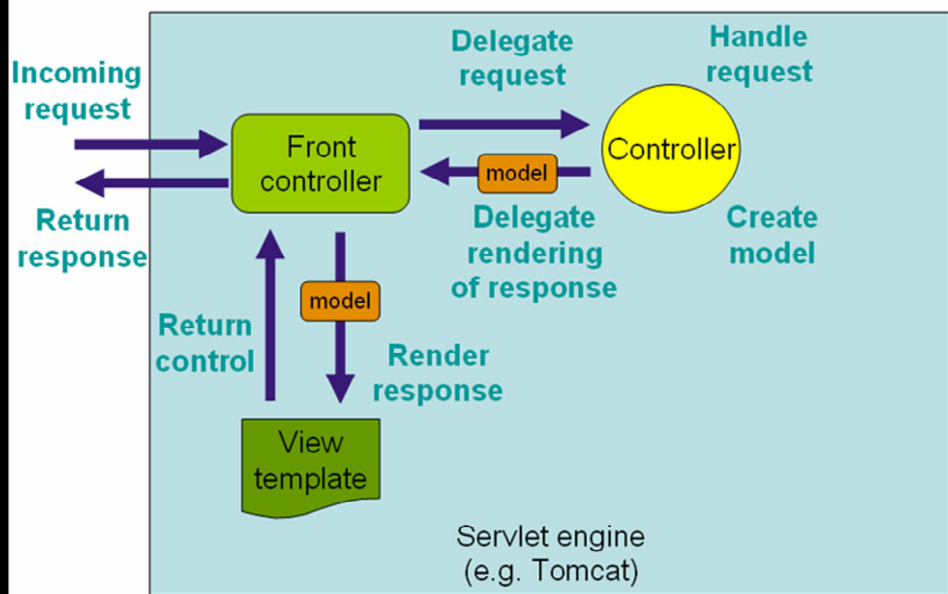
Model state

The model represents the state of the application. The business objects update the application state. ActionForm bean represents the Model state at a session or request level, and not at a persistent level. The JSP file reads information from the ActionForm bean using JSP tags.

View

The view is simply a JSP file. There is no flow logic, no business logic, and no model information -- just tags. Tags are one of the things that make Struts unique compared to other frameworks like Velocity.

Spring MVC



1-Mar-23 5:15 PM

11

State DP

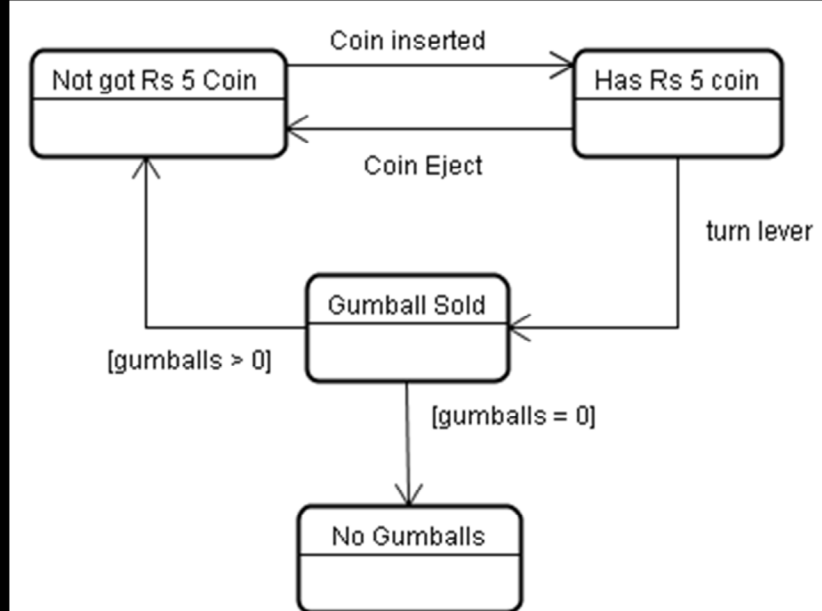
1-Mar-23 5:15 PM

12

Problem

- We have a gumball machine.
 - People put a coin in it to get a big ball of chewing-gum.
 - Simulate the function of this machine.

State diagram



Code for this Class?

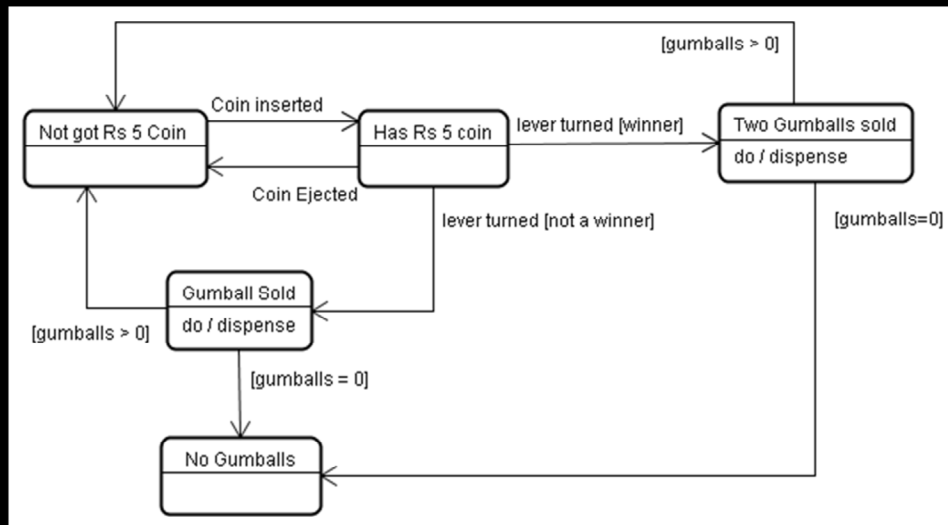
GumBallMachine

- + coinInserted() : void
- + leverTurned() : void
- + dispense() : void
- + ejectQuater() : void

Problem

- Now, we have a change request.
 - One out of the ten customers get 2 gumballs instead of 1. These 10% customers are the lucky ones.
 - How does the state diagram change?
 - How does the code change?

State diagram



1-Mar-23 5:15 PM

17

The code does not adhere to Open Closed principle.

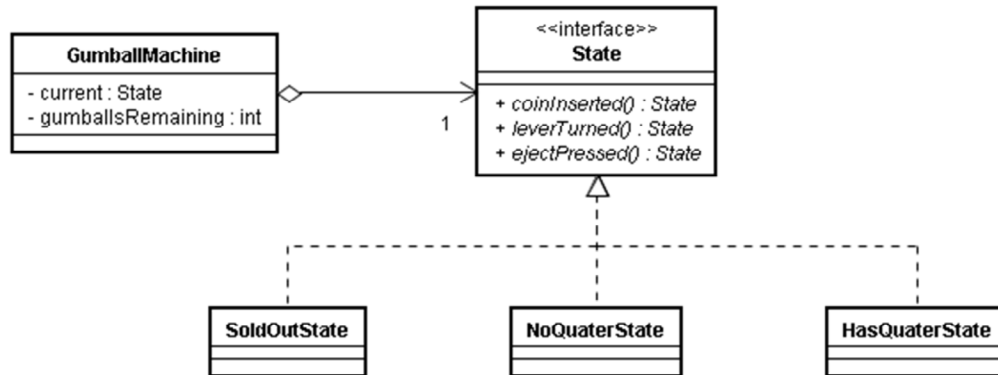
State transitions are not explicit. They are buried in the middle of conditional statements.

We have not encapsulated anything that varies.

Further changes are probable. They will need existing code to be changed.

The design is not object oriented.

State DP

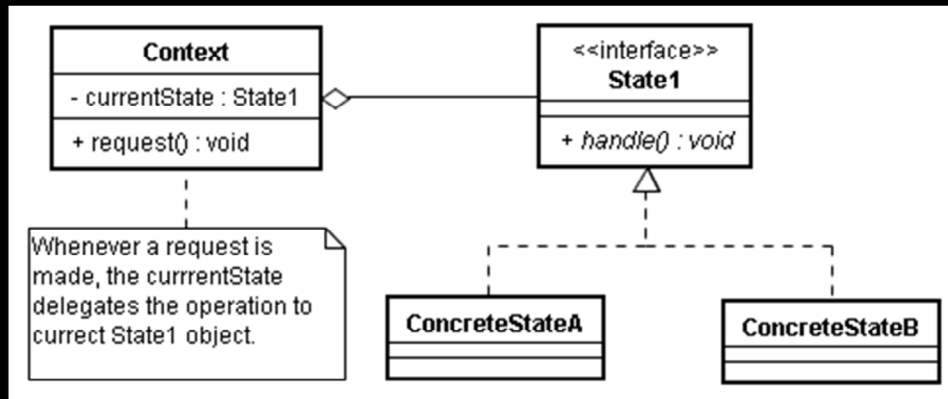


1-Mar-23 5:15 PM

18

Code example

State DP



1-Mar-23 5:15 PM

19

The context

is of interest to Clients

maintains an instance of a concreteState subclass that defines the current state.

It switches from one state to another during its lifetime, in order to produce different behavior from the same method call(s).

The State

defines an interface for encapsulating the behavior associated with a particular state of the Context.

The ConcreteState subclasses

Each subclass implements a behavior associated with a state of the context.

=====

It allows an object to have many different behaviors that are based on its current internal state.

Clients never interact directly with the State objects.

It eliminates the necessity for a set of long, look-alike conditional statements scattered through the program's code.

It makes the state transitions explicit.

Without this DP, some variable value implicitly determines the state of object.