# Padrões de Projeto de Software Orientados a Objetos

## Tecnologia em Análise e Desenvolvimento de Sistemas

Paulo Mauricio Gonçalves Júnior

Instituto Federal de Educação, Ciência e Tecnologia de Pernambuco

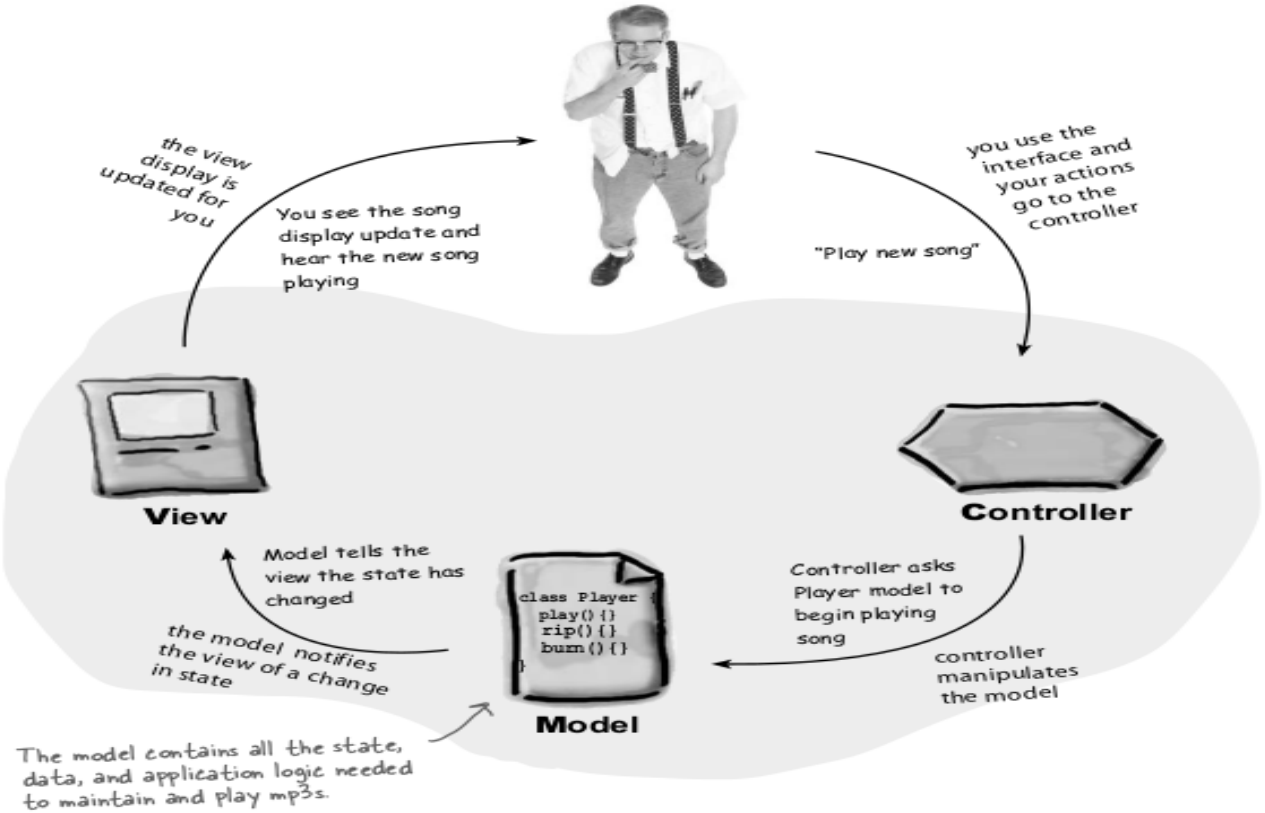
8 de junho de 2018

Parte II

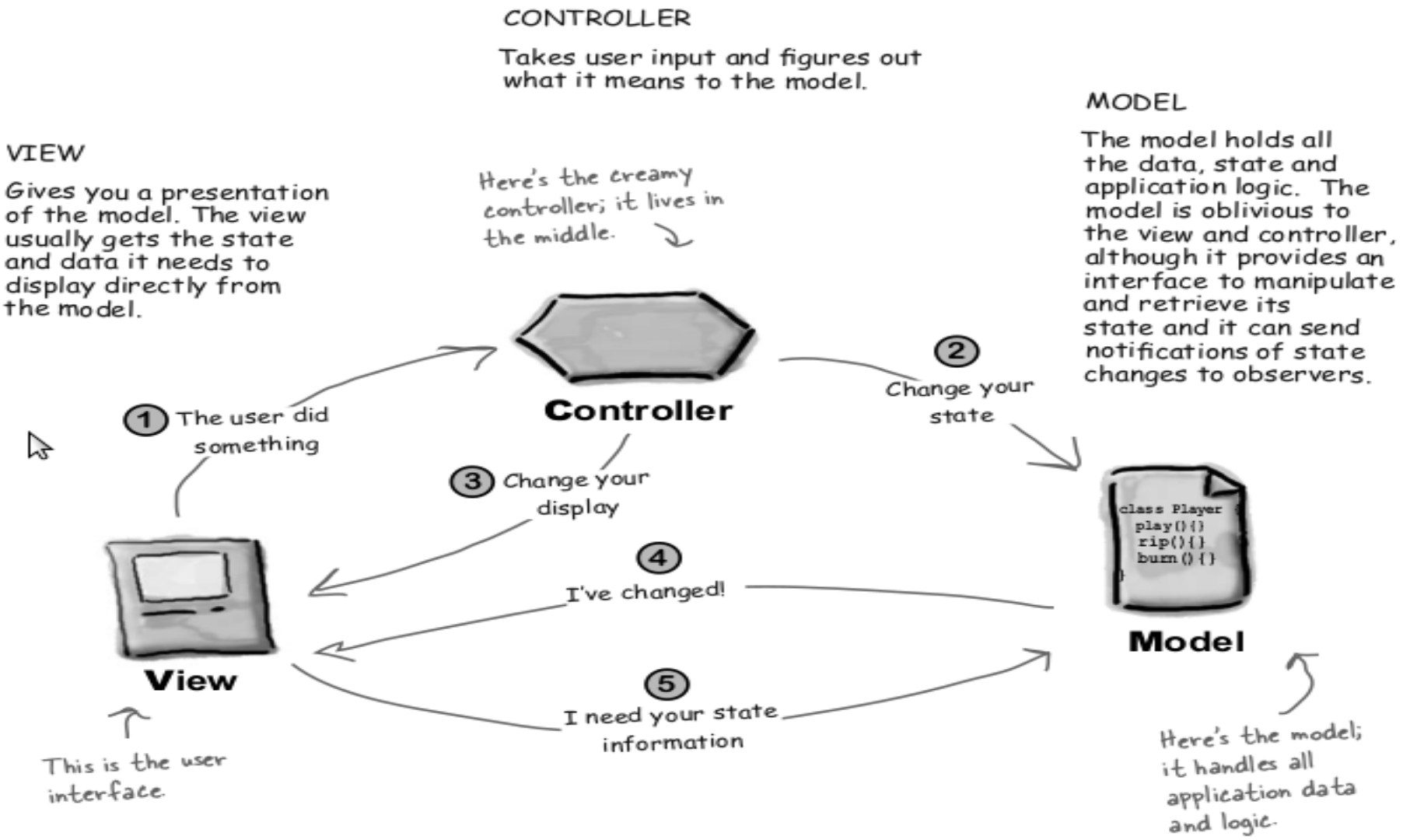
# MVC – Model View Controller

# MVC I

- MVC é um padrão composto.
- Imagine um tocador de MP3 portátil.



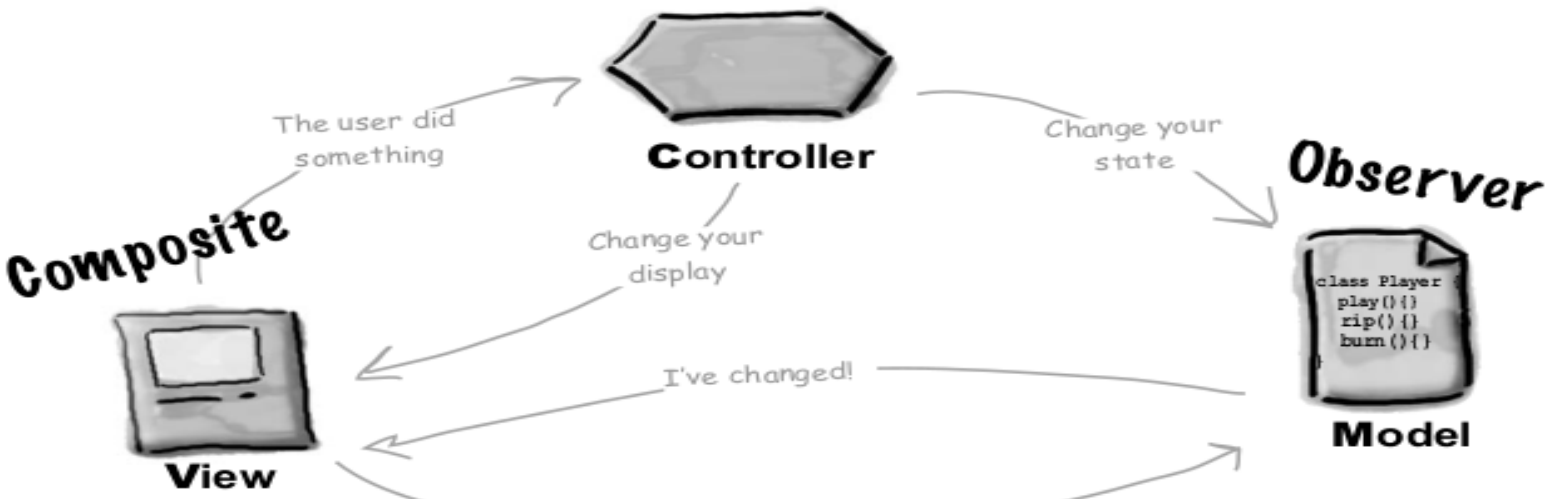
# MVC II


CONTROLLER
Takes user input and figures out what it means to the model.
MODEL
The model holds all the data, state and application logic. The model is oblivious to the view and controller, although it provides an interface to manipulate and retrieve its state and it can send notifications of state changes to observers.
VIEW
Gives you a presentation of the model. The view usually gets the state and data it needs to display directly from the model.
Here's the creamy controller; it lives in the middle.
Controller
1 The user did something
2 Change your state
3 Change your display
4 I've changed!
5 I need your state information
View
Model
class Player
play(){}
rip(){}
burn(){}
This is the user interface.
Here's the model; it handles all application data and logic.

# MVC III

## Strategy

The view and controller implement the classic Strategy Pattern: the view is an object that is configured with a strategy. The controller provides the strategy. The view is concerned only with the visual aspects of the application, and delegates to the controller for any decisions about the interface behavior. Using the Strategy Pattern also keeps the view decoupled from the model because it is the controller that is responsible for interacting with the model to carry out user requests. The view knows nothing about how this gets done.




The display consists of a nested set of windows, panels, buttons, text labels and so on. Each display component is a composite (like a window) or a leaf (like a button). When the controller tells the view to update, it only has to tell the top view component, and Composite takes care of the rest.

The model implements the Observer Pattern to keep interested objects updated when state changes occur. Using the Observer Pattern keeps the model completely independent of the views and controllers. It allows us to use different views with the same model, or even use multiple views at once.

# MVC2 I

- MVC foi adaptado para funcionar no modelo cliente/servidor, chamado MVC2.
- Ele permite a separação de responsabilidades na produção.