

# Programming Paradigms, Software Architectural Patterns, and MVC



---

## Programming Paradigms

A *paradigm* can be defined as a set of assumptions, concepts, values, and practices that constitutes a way of viewing reality for the community that shares them, especially in an intellectual discipline.

A *programming paradigm* is often associated with a particular programming language.

- The choice of the language often dictates the paradigm, and vice-versa.

---

## Notable Programming Paradigms

**Functional Programming** (1930s to present): all about the evaluation of mathematical functions. Avoids state changes and mutable data.

**Imperative Programming** (1950s to 1970s): defines computation as statements that change a program state.

**Procedural Programming** focused on *procedures*, also known as “routines”, “subroutines” and “functions”.

**Structured Programming** focused on making programs easier to write, debug, and understand.

---

## Notable Programming Paradigms (cont.)

**Declarative Programming** defines computational logic without defining its control flow.

**Object-Oriented Programming** organizes programs as objects: data structures with attributes and methods together with their interactions.

In **Automata-Based Programming**, a program, or part of a program, is treated as a model of a finite state machine or similar formal automaton.

---

## Event-Driven Programming

In **Event-Driven Programming**, control flow is determined by events, such as input from sensors or user actions (mouse clicks, key presses, etc.) or messages from other programs or threads

- The notion is that the application sits, waiting for input from the user — which can come from many directions.
- *Event-driven programming is the dominant paradigm* for graphical user interfaces and other applications that are centered on performing certain actions in response to user input.

---

## Software Architectural Patterns

**Software Architectural Patterns** are reusable approaches or solutions to problems in software design that show up frequently.

- They act like templates on which you base your software
- Software architectural patterns are conceptual, such as “peer-to-peer networking”.
- Software architectures are actual implementations of a pattern, such as TCP/IP.

---

## SW Architectural Pattern: Model-View-Controller

*Model-View-Controller* (MVC) is a software architectural pattern for implementing user interfaces.

- It follows the event-driven paradigm for control flow.
- It divides an application into three interconnected parts in order to separate internal structures from the way they're presented to the user.

### **Model**

- Consists of objects that encapsulate the data specific to the application
- Defines the logic and computations that manipulate and process that data.

---

## SW Architectural Pattern: Model-View-Controller

### **View**

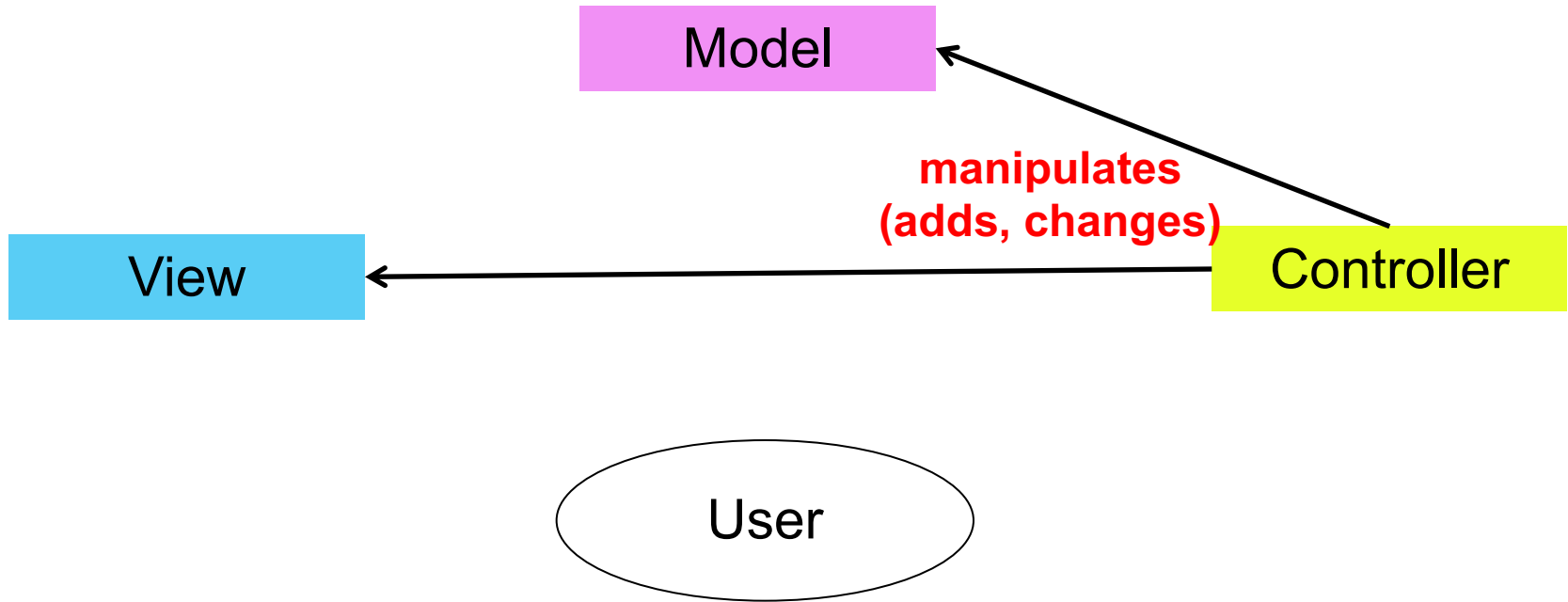
- Any object in an application that users can see. Main purpose: to display data from the model objects and enable editing of that data
- View objects know how to draw themselves, and can respond to user actions
- Provide consistency between applications
- Multiple views of the same information is possible; for example, a bar chart for management and a table view for finance.

### **Controller**

- Acts as an intermediary between view objects and model objects
- Accepts input and converts it to commands for the model or view
- Can also perform setup and coordinating tasks for an application and manage the life cycles of objects



## MVC: Controller

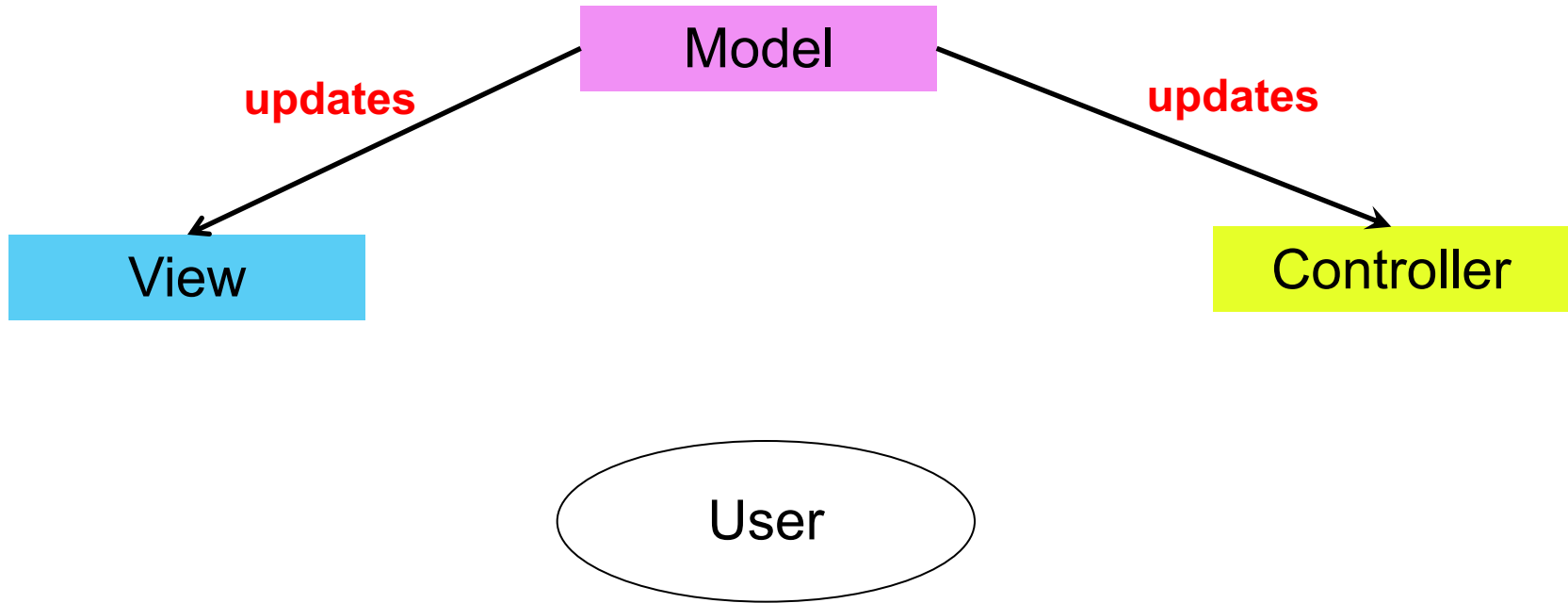


### Controller:

- Interprets user actions made in view objects and communicates changes to the model to update the model's state
- When model objects change, controller communicates changes to the view to change the view's presentation

---

## MVC: Model

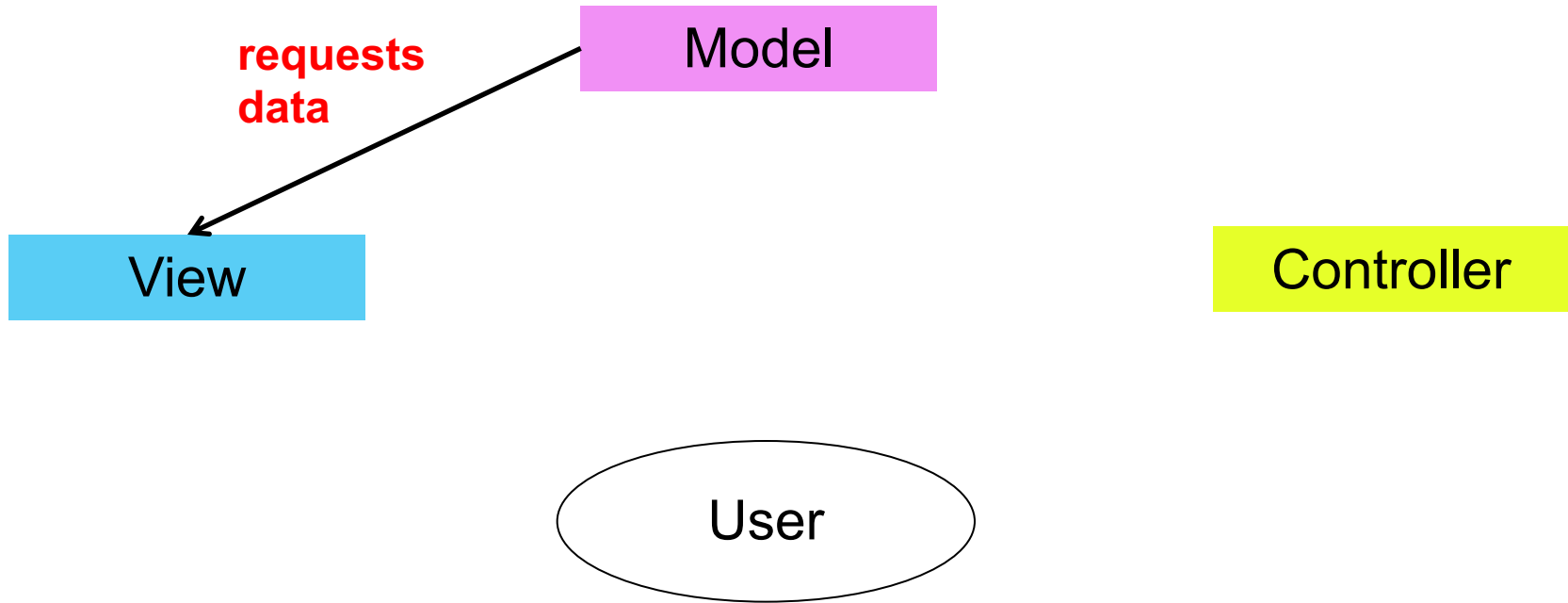


### Model:

- Tells the View when there has been a state change, causing the output from the View to be updated.
- Tells the Controller when there has been a state change, causing the Controller to change the set of commands available.

---

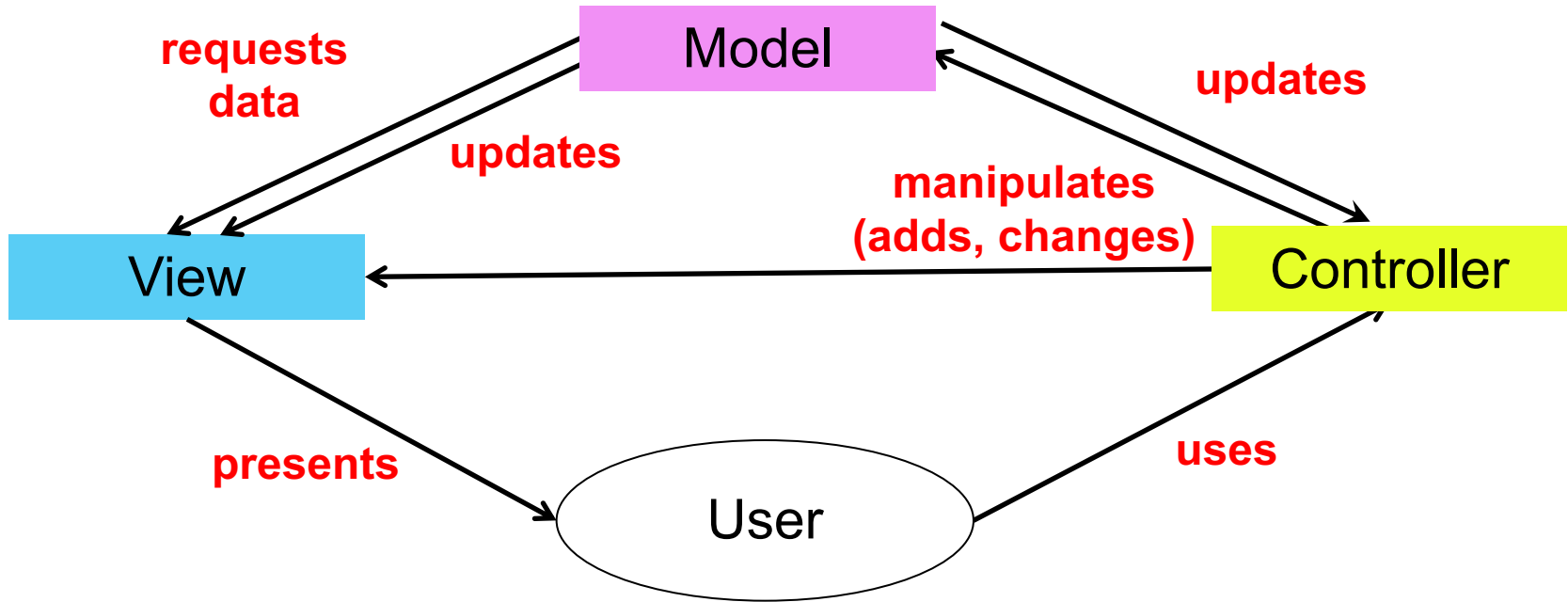
## MVC: View



### View:

- Requests information from the model that it uses to generate output to the user

## MVC (cont.)



Note that the user never interacts directly with the model.

## Model-View-Controller (cont.)

Model-View-Controller is often used in Web applications and Mobile applications.

- The View presents a window with controls to the user
- Any data that the user enters is sent to the Controller; for example, selecting an item such as “Notifications”.
- The Controller updates the Model (i.e., stores the data in a data structure, updates objects, etc.)
- The Model tells the View to display a different window
- The Model also tells the Controller that the controls from the previous window no longer work, and there are new ones to pay attention to.

